Theory and Implementation of Puzzles with GAP

Master Thesis

BY

Angelou Areti-Christina

University of Thessaly
Department of Computer and Electrical Engineering

Supervisor: Akritas Alkiviadis

Co-supervisor: Daskalopoulou Aspasia

Co-supervisor: Stamoulis Georgios

Volos, March 2016

Abstract

There are many puzzles, that have captured the imagination of millions, are accessible to all age levels, are challenging, yet satisfying, and are so mathematically rich. The Rubik's Cube is one such puzzle, others examples include the 15-puzzle, TopSpin, Hungarian Rings. All these puzzles have a common theme: the pieces of the puzzle can be rearranged, and the goal is to restore the pieces back to their original positions. In other words, the pieces are permuted. The legal moves that one is allowed to use is forced by the design or construction of the puzzle. Puzzles of this type known as permutation puzzles. There are two types of permutations associated with a puzzle, the permutation describing the puzzles current position and the permutation corresponding to a move sequence applied to the puzzle. The permutation describing the puzzles position is a composition of the permutations corresponding to the puzzle moves which takes the puzzle from the solved state to that position. That is, multiplying the permutations corresponding to the moves, should give us the permutation of the resulting position. These permutations of puzzles position, can be decomposed as a product of 2-cycles or 3-cycles and this decomposition can be connected to the solvability of puzzles.

In this thesis we will use these various puzzles and our goal is to uncover mathematics behind these puzzles. Furthermore we want to see how we can model these puzzles mathematically. Especially there is a full analysis of the 15-Puzzle, where a solvability criteria and strategy for solution is presented and also a complete solution of the 15-Puzzle is presented using GAP (version 4.7.8), a computer algebra system for computational discrete algebra.

Acknowledgments.

Foremost I would like to express my sincere gratitude to my supervisor, professor Alkiviadis Akritas, for his overall guidance, support and motivation in all the time of implementing this thesis and especially for his confidence in me. I would like to express my deeply appreciation to him for his help and the time he disposed.

Moreover I would also like to sincerely thank my co-supervisors, professor Aspasia Daskalopoulou and professor Georgios Stamoulis for their willingness to supervise my thesis and their help on this and other projects through my studies.

Last but not least, I would like to express my special thanks to my family and Marios for their endless support and love that showed to me all these years. Their encouragement and trust led me this far.

Table of contents

1 Introduction to GAP

GAP (Groups, Algorithms and Programming) is a computer algebra system for computational discrete algebra with particular emphasis on computational group theory. GAP provides a programming-language, a library of thousands of functions implementing algebraic algorithms written in the GAP language as well as large data-libraries of algebraic objects. GAP is used in research and teaching for studying groups and their representations, rings, vector spaces, algebras, combinatorial structures, and more. The system, including source, is distributed freely and it can be easily modified and extended for special use [1].

GAP was developed at Lehrstuhl D für Mathematik (LDFM), RWTH Aachen, Germany from 1986 to 1997. After the retirement of J. Neubüser from the chair of LDFM, the development and maintenance of GAP was coordinated by the School of Mathematical and Computational Sciences at the University of St Andrews, Scotland. In the summer of 2005 coordination was transferred to an equal partnership of four "GAP Centres", located at the University of St Andrews; RWTH Aachen; the Technische Universität Braunschweig; and Colorando State University at Fort Collins [2].

GAP provides Mathematical capabilities accessible through: i) a large library of functions, containing implementations of various algebraic algorithms, ii) separate packages of additional functions for specialized purposes which can be used like library functions, iii) data-libraries containing large classes of various algebraic objects that are accessible by using GAP commands. GAP also provides a programming language, also called GAP, which is interpreted and can be compiled. It can be used interactively at the keyboard or to write programs to be saved and then executed. Such programs can easily be modified and rerun. The language has many features such as: Pascal-like control structures, automatic memory management including garbage collection, streams, flexible list and record data types, built-in data types for key algebraic objects and automatic method-selection building on a mechanism for automatically choosing the highest ranked method for a certain operation, depending on the current state of all its arguments, so that GAP objects representing mathematical objects may gain knowledge about themselves during their lifetime resulting in better methods being chosen later on. Moreover, GAP provides an interactive-environment that supports in particular line-editing (e.g. tab completion), break-loops for debugging, further debugging-and-profiling facilities for GAP programs, online-help (i.e. online access to the manuals), a graphical-user-interface for GAP and further GAP interface programs provided by users[1].

GAP has a kernel written in C. It implements the GAP language, an interactive environment for developing and using GAP programs, memory management, and fast versions of time critical operations for various data types. All the rest of the library of functions is written in the GAP language. Packages are mainly written in the GAP language, but some also involve standalones. Some packages provide links to other systems[1].

The GAP system will run on any machine with a UNIX-like or recent Windows or Mac OS X operating system and with a reasonable amount of RAM and disk space. The current version is GAP-4 and it can be obtained at no cost by following the download instructions: http://www.gap-system.org/Download/index.html

2 Introduction to permutation puzzles

Rubik's Cube is probably one of the most well known puzzles to date. It is estimated that over 350 million have been sold since it's creation by Ern-o Rubik around 1980. What has made it so popular is not certain. Perhaps it looks seemingly innocent, then once a few sides have been twisted, and the colours begin to mix, the path back home is not so easy to see. The more you twist it the further you seem to be taken away from the solution. Perhaps it is that the number of ways to mix up the cube seems endless. Or perhaps to others, it doesn't seem endless at all. Despite the reasons for its appeal, it has become one of the most popular puzzles in history.

It is rare to find a puzzle, or toy, that has captured the imagination of millions, is accessible to all age levels, is challenging, yet satisfying, and is so mathematically rich. The Rubik's Cube is one such puzzle. Others examples include the 15-puzzle, TopSpin, Hungarian Rings, and Lights Out. This is the theme for all these puzzles. There is a certain stage in solving the puzzle where a simple strategy, and trial and error, can't get you any further. This is typically referred to as the end-game for the puzzle. For Rubik's Cube the end-game is usually when two layers are solved, and the last layer remains, while for 15-puzzle the end-game is usually when all pieces are solved and the pieces 14 and 15 remains. It is understanding the end-game of these puzzles that mathematics becomes such a useful tool.

It turns out that one area of mathematics that has had an impact on all areas of science, and has even popped up in art, is the area called group theory. Often referred to as the language of symmetry, group theory has led to many new discoveries in theoretically physics, chemistry and mathematics itself. It underlies the techniques in cryptography (sending private information over public channels), and coding theory (digital communications, digital storage and retrieval of information).

In this thesis we will use these various puzzles and our goal is to uncover mathematics behind these puzzles. Furthermore we want to see how we can model these puzzles mathematically. Especially there is a full analysis of the 15-Puzzle and also a complete solution of the 15-Puzzle using GAP is presented.

This thesis depends on the theory background stated on David Joyner's book "Adventures in Group Theory"[3] and on Jamie Mulholland's lectures [4] of Permutations Puzzle Course at Simon Fraser University, which the reader can consult for more details.

The rest of the paper is organised as follows:

In this section (section 2) there is a briefly description of the puzzles that will be investigated in this thesis and also the definition of permutation puzzles is given.

In section 3 there is a presentation of the basic terminology and notation from the set theory which will form the foundation for our mathematical investigations into puzzles. Also there are examples of sets using GAP.

In section 4 there is an introduction of some terminology and notation about rearrangements of objects. In particular, the definition of a permutation is given, which is the main object for the investigation of these puzzles. There is also the definition of permutation multiplication, inverses, exponents and order and also examples of permutations using GAP.

In section 5 there is the representation of the permutation puzzles, which are investigated in this thesis, by a permutation. There are two types of permutations associated with a puzzle, the permutation describing the puzzles current position and the permutation corresponding to a move sequence applied to the puzzle. Also there are examples of the representantion of the puzzles using GAP.

In section 6 there is a presentation of products of traspositions of permutations. There is the description of how permutations could be presented as products of 2-cycles and 3-cycles and also the parity theorem and the proof of this theorem is presented. Moreover the solvability of Swap puzzle is investigated.

In section 7 there is a full analysis of the 15-puzzle, with a solvability criteria about whether a given scrambling of the puzzle is solvable. Also the strategy for solving the puzzle and the solution of the 15-puzzle using GAP (version 4.7.8) are presented.

2.1 A collection of puzzles

We begin by briefly describing the puzzles we will be investigating in this thesis. One thing to observe is that all puzzles have a common theme: the pieces of the puzzle are rearranged, and the goal is to return the pieces to their proper (original) arrangement.

2.1.1 A basic game, Swap

Imagine a set of objects laid out in front of us and ordered in some way. This puzzle can be played with any number of objects, but the more objects that are used the more challenging it becomes.

It doesn't matter what the objects are, they could all be different, or some could be the same. We will start with 5 distinct objects, and for simplicity we will just take the objects to be the numbers 1, 2, 3, 4, and 5.

This arrangement, where the numbers appear in order from left to right, is called the home position or solved state. Since, as we'll see shortly, we will be moving the numbers around the boxes so it will be nice to have a little reminder of whose home is whose. We do this by putting a little number in the top left corner of each box. Figure 1 shows the objects laid in solved state:

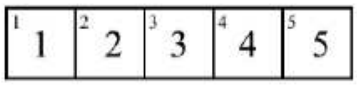


Figure 1. Solved state of Swap with 5 objects. Boxes labeled.

The way this puzzle is played is the following. First the numbers are randomly arranged in the boxes. Then using only legal moves, one tries to move the numbers back to their home positions (i.e. return the puzzle to its solved state). This is where different version of the puzzle can be created. For now, let us simply say there is one type of legal move called a swap, and it consists of picking any two boxes and swapping the contents (the numbers in large font).

Example 2.1 *Considering the starting position in Figure* 2*, our goal is to return the numbers to the solved state using only legal moves. We notice that objects* 2 *and* 4 *are in the correct positions (correct boxes).*

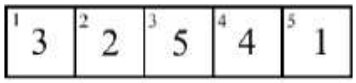
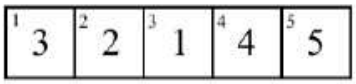
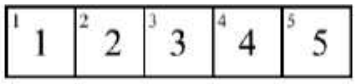


Figure 2. Staring position for Example 2.1.

This is where the little numbers in the top left corners come in handy. As for the numbers $1, 3$ *and* 5*, we need to move these to their correct positions. Since the legal moves consist of swapping the contents of two boxes at a time, we'll focus first on getting* 5 *into it's correct position. To do this we swap the contents of boxes* 3 *and* 5*, since object* 5 *is in box* 3*.*



Now $2, 4$ *and* 5 *are in the correct positions. Lastly we swap the contents of boxes* 1 *and* 3 *and solve the puzzle.*



This is a basic puzzle and we will have this as our starter puzzle. Every puzzle we will investigate will just be some variation of Swap. Either we will increase the number of objects, or we will change the legal moves. We now consider some possible variations of the puzzle.

Variations of Swap: There are many ways to vary this puzzle. One way is to increase the number of objects that are used. Here we used 5, but we could use 10, 20, 48, or any number we wish.

Another way to vary the puzzle is to choose a different collection of legal moves. Our legal moves consisted of swapping the contents of two boxes. Instead we could have stated that legal moves only consisted of swapping the contents of box 1 and any other box. If this was the case then the solution in Example 2.1 would be illegal, since started by swapping the contents of boxes 3 and 5, which is a move that doesn't involve box 1.

We could also extend the notion of a legal move beyond "swaps". For instance we could use only moves of the form: pick three boxes and cycle the contents either to the right (clockwise) or to the left (counterclockwise).

For example, considering 6 objects in Figure 3, we could cycle the contents of boxes 2, 3 and 5 to the left (other boxes are shaded to allow us to focus on what is changing).

Figure 3. Legal move variation: 3-cycle to the left.

We can now describe the puzzle **Swap**, including all possible variations.

Rules of Swap (*n* objects):

Let T be a set of n objects, and suppose the objects have been ordered in some way. For example, the objects can be the numbers 1 to n and the ordering could be their natural ordering from smallest to largest written from left to right. When the objects are in their proper order, we say they are in their home positions and the puzzle is in the solved state. Let M be a collection of legal moves.

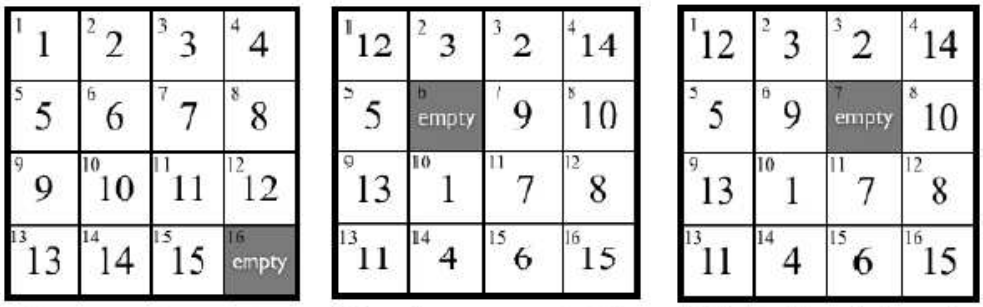
1. **Puzzle Start:** Randomly arrange the numbers 1 through n from left to right.

2. **Puzzle Play:** Using only legal moves (i.e. moves in M), return the puzzle to the solved state.

Stated here in its most general form we'll see that most Rubik's Cube-like puzzles are just variations of Swap. Of course, this connection with Swap doesn't make the Rubik's Cube any easier to solve, but it will provide us a way to investigate and understand the cube and other puzzles.

2.1.2 The 15-puzzle

The 15-puzzle consists of a 4×4 grid with numbered tiles from 1 to 15 placed in the grid. The space where the 16 tile would go is left empty (Figure 4a).



(a)The 15-puzzle in its solved state.

(b)A random arrangement of the 15-puzzle.

(c)Obtained from 4b by moving the tile in box 7 to box 6.

Figure 4. The 15-puzzle

The little numbers in the top left corner of each box are not present on any of the manufactured puzzles, nor are they present on the software version of the puzzle. But these little numbers proved to be so handy in reminding us where each tiles home position is in Swap that we'll use them here to.

The object of this puzzle is to randomly arrange the tiles on the grid, and then through a sequence of legal moves which consist of sliding a tile into the empty space (which results in the empty space moving around the board), one tries to return all tiles to their home positions.

Current manufactured versions of the puzzle are assembled in such a way that the pieces are not removable. There is a tongue-and-grove design which allows the pieces to slide around but doesn't allow them to be removed. However, the original versions of the puzzle (manufactured in the 1880's) consisted of removable wooden pieces. This little difference in puzzle constructions has significant impact on the ability to solve the puzzle. This puzzle started a craze that swept across the nation, and across the world, from January to April of 1880. All the fuss was centred around the fact that after randomly putting the wooden blocks back into the box, solving the puzzle seemed to take you to one of two places: either you solved it completely, or you got every number in its correct position except the 14 and 15 were switched (Figure 5). In the case when the last two tiles were switched it seemed the puzzle wasn't solvable.

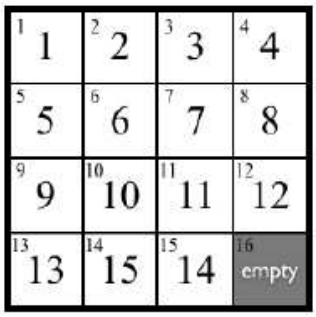


Figure 5. The 13-14-15 Problem. Can the puzzle be solved by starting from this position?

In section 7 we will investigate whether this arrangement of the puzzle is solvable, as well as come up with a strategy for solving the puzzle in general and the solution of the puzzle with GAP.

2.1.3 The Oval Track Puzzle (or TopSpin™)

The TopSpin puzzle was manufactured by Binary Arts. It was invented by Ferdinand Lammertink, and patented on 3 Oct 1989, US 4,871,173 [6]. The puzzle consists of 20 numbered round pieces in one long looped track (Figure 6). We can slide all the pieces around the loop. There is also a turntable in the loop (this is the purple circle which contains disks 1 through 4 in Figure 6), which can rotate any four adjacent pieces so that they will be in reverse order. This in effect swaps two adjacent pieces and the two pieces on either side of them. The aim is of course to mix up the ordering of the pieces, and then place the pieces back in numerical order (as show in Figure 6).[5]

This puzzle became a North American classic with over a million copies sold. Nowadays though, the only place that this puzzle is available in its physical form is on ebay.

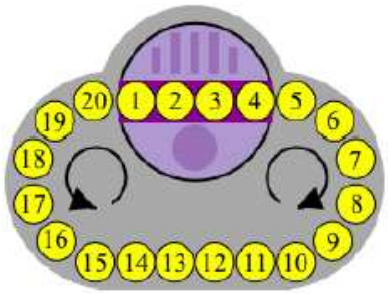
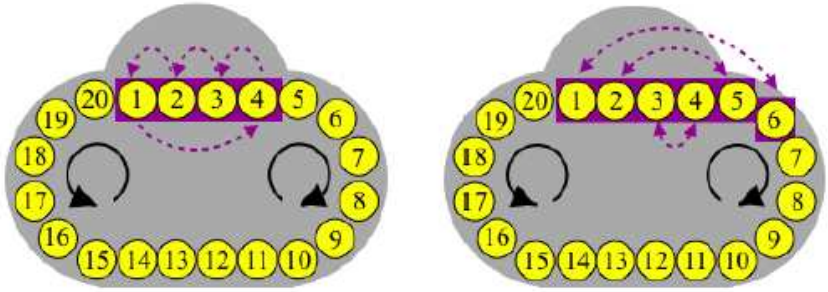


Figure 6. The TopSpin Puzzle in its solved state.

Variations of Oval Track: The name Oval Track has been given to a more general version of the puzzle, one that is updated for the digital era. When one moves away from having to construct a physical puzzle, and instead creates a virtual puzzle, then a whole new world of possible moves is available. For example, Figure 7 shows two variations of the puzzle. The turntable move in the original TopSpin puzzle is now replaced with the move indicated by the purple dashed lines. For

instance, the new turntable move for the puzzle in Figure 7a moves the disk in spot 4 to spot 3, the disk in spot 3 to spot 2, the disk in spot 2 to spot 1, and takes the disk in spot 1 to spot 4. Another version of the turntable move involving 6 disks is given in Figure 7b. Variations of this puzzle are limited only by our imagination.



(a)One variation of the turntable move

(b)Another variation of the turntable move

Figure 7. Variations on the Oval Track Puzzle

As usual, it will be convenient to indicate the home positions of the disks. So we put little numbers around the track indicating the number of the disk that should be in that position for the puzzle to be solved (Figure 8).

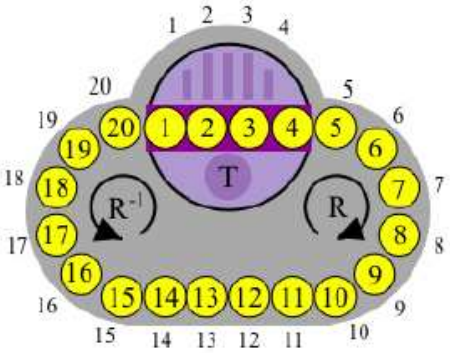


Figure 8. The TopSpin Puzzle with home positions labeled, and move notation

Oval Track Notation: A clockwise rotation of numbers around the track, where each number moves one space, is denoted by R, a counterclockwise rotation is denoted by R^{-1}. A rotation of the turntable is (in general, an application of the *follow the arrows* move) is denoted by T. Figure 8 provides a visual summary of this notation.

2.1.4 Hungarian Rings

The Hungarian Rings puzzle consists of two intersecting rings made up of a number of coloured balls. The rings of balls intersect at two places, so they share two of the balls. Each ring of balls can be rotated, so the balls can be mixed. The aim is to mix up the balls, and then place the balls back together so the colour form a continuous sequence (as show in Figure 9).

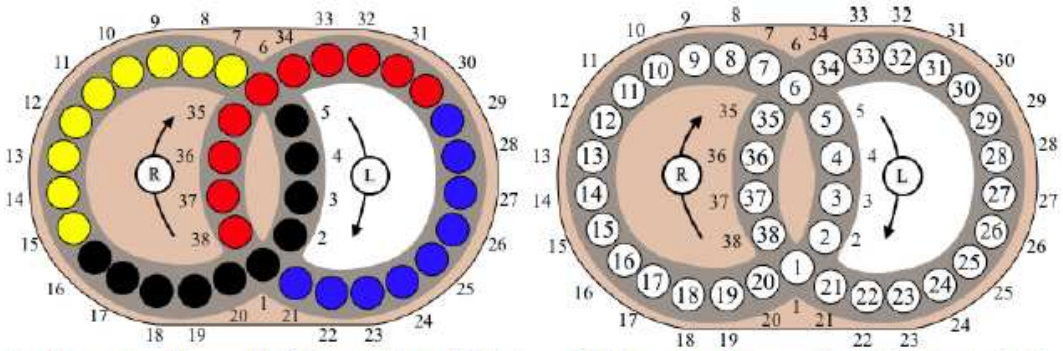
There are 38 balls of four colours: two colours have 9 balls (yellow and blue) and two colours have 10 balls (black, red). There are 4 balls between the intersections of the rings.[8]

Endre Pap, a Hungarian engineer, invented a flat version with two rings which was marketed as the Hungarian Rings. The idea was not entirely new, as there is an 1893 patent for it by William Churchill, filed on May 28 1891, granted on October 24, 1893. (In Jaap's Puzzle page there is a copy of the patent.[7])

Figure 9. Hungarian Rings in its solved state. (manufactured 1982)

To study this puzzle we will temporarily ignore colours, and instead assign a number to each ball (as shown in Figure 10b). We'll also indicate the home position of each ball by putting little numbers along the outside of the track. In effect we will study the puzzle of rearranging the numbers 1 through 38 on the two rings. In some sense this is a more difficult puzzle than the colour version of the puzzle simply because in the colour version there are really only 4 distinct balls, whereas in the number version there are 38 distinct balls and each one has only one home position. However, as we'll see, the added complexity inherited by using numbers, rather than colours, is not so bad, and the benefits to understanding the puzzle are numerous.

Hungarian Rings Notation: A clockwise rotation of the balls in the **right-hand ring**, where each ball moves one space around the track, is denoted by R, a counterclockwise rotation is denoted by R^{-1}. A clockwise rotation of the balls in the **left-hand ring**, where each ball moves one space around the track, is denoted by L, a counterclockwise rotation is denoted by L^{-1}. Figure 10 provides a visual summary of this notation.



(a) Hungarian Rings with home positions labeled by numbers.

(b) Hungarian Rings with numbers instead of colours.

Figure 10. Hungarian Rings with disks labeled by numbers

2.1.5 Rubik's Cube

This is probably the most well known mechanical puzzle. It was invented by Ernö Rubik in Hungary around 1974, and the patent was filed 30 January 1975, HU 170062 [9]. Eventually it was produced and marketed by Ideal Toys in the early 1980s. It is quite possibly the most popular toy to have ever been manufactured, and many copycat cubes were made. It is estimated that there have been over 350 million cubes manufactured since 1980, and it is still being manufactured today.

The Rubik's Cube is a cube which is built from smaller cubes where there are 3 cubes along an edge, i.e. a $3\times3\times3$ cube. The 9 pieces on each face can rotate, which rearranges the small cubes at that face. The six sides of the puzzle are coloured, so every corner piece shows three colours, every edge piece shows 2 colours, and every face centre only one (as shown in Figure 11).[10]

(a) View of front (red), right (yellow) and up (blue) faces.

(b) View of back (orange), left (white) and down (green) faces.

Figure 11. The $3\times3\times3$ Rubik's Cube with classic colouring scheme: blue opposite green, red opposite orange, white opposite yellow.

Turning a face does not change the face centres (this is because twisting the face centres is not a visible change of pattern, if however, there was an image rather than a solid colour on the face then this would not be the case anymore) so these can be considered already solved. The other pieces have to be placed correctly around them. This is a particularly important observation because it implies the following:

The colour of the centre piece of any face defines the only colour to which that face of the cube can be restored.

Cube Terminology & Notation: When playing with the cube the pieces begin to move all around. Since there are so many moving parts of the cube, it will be convenient to have some terminology to describe each piece, and its placement in the cube. It will also be convenient to have some notation for basic movements to aid in communication with one another. Of course, a good choice of notation can bring mathematics into the picture as well, as we will soon see. The notation we use was first introduced by David Singmaster in the early 1980's [12], and is the most popular notation in use today.

We may label the 6 sides as $\boldsymbol{f}, \boldsymbol{b}, \boldsymbol{r}, \boldsymbol{l}, \boldsymbol{u}, \boldsymbol{d}$ for *front, back, right, left, up* and *down*.

The cube is made up of 26 smaller cubes called **cubies**. These are the ones that are visible, there actually isn't a 27^{th} cube in the middle, but instead a mechanism that allows things to twist and turn. The cube has 6 sides, or **faces**, each of which has $3\cdot3=9$ **facets**. Think of a facet as just one of the little coloured stickers. There are 54 facets in total for the $3\times3\times3$ Rubik's cube. The cubes split up into three types: **centre cubies** (having only one facet), **edge cubies** (having two facets), **corner cubies** (having three facets), as shown in Figure 12.

(a) 6 centre cubies and mechanism

(b) 12 edge cubies

(c) 8 corner cubies

Figure 12. A disassembled Rubik's Cube showing the cubies.

Each face of the cube is made up of a slice of 9 cubies that share a facet with the face. The face, along with all of the 9 cubies in the slice, can be rotated by 90 degrees clockwise (viewing the face straight-on). We denote this move by uppercase letter of the name of the face. For example, F denote the move which rotates the front face by 90 degrees clockwise. Table 1 [4] describes a complete description of cube moves and notation[3].

We call the space which a cubie can occupy a cubicle. As the pieces move around, the cubies move from cubicle to cubicle, and the facets move to the locations previously occupied by other facets. In order to solve the puzzle each cubie must get restored to its original cubicle, we call this its home location, and the facets must also be correctly positioned, we call this the cubies home orientation. Once all cubies are in their home positions and home orientations the puzzle will be solved.

Table 2 [4] summarizes the terminology introduced here.

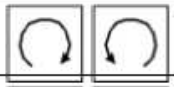
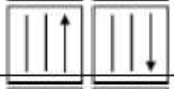
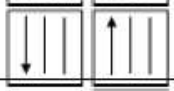
notation (Singmaster)	pictorial (view from front)	description of basic move (clockwise/counterclockwise refers to viewing the face straight-on)
F , F^{-1}		F = quarter turn of **front** face in the **clockwise** direction. F^{-1} = quarter turn of **front** face in the **counterclockwise** direction.
B , B^{-1}		B = quarter turn of **back** face in the **clockwise** direction. B^{-1} = quarter turn of **back** face in the **counterclockwise** direction.
R , R^{-1}		R = quarter turn of **right** face in the **clockwise** direction. R^{-1} = quarter turn of **right** face in the **counterclockwise** direction.
L , L^{-1}		L = quarter turn of **left** face in the **clockwise** direction. L^{-1} = quarter turn of **left** face in the **counterclockwise** direction.
U , U^{-1}		U = quarter turn of **up** face in the **clockwise** direction. U^{-1} = quarter turn of **up** face in the **counterclockwise** direction.
D , D^{-1}		D = quarter turn of **down** face in the **clockwise** direction. D^{-1} = quarter turn of **down** face in the **counterclockwise** direction.
F^2, B^2, R^2, L^2, U^2, D^2 denote the corresponding *half-turn* of the face. Since a clockwise half-turn is equivalent to a counterclockwise half-turn then $F^2 = F^{-2}$, $B^2 = B^{-2}$, $R^2 = R^{-2}$, $L^2 = L^{-2}$, $U^2 = U^{-2}$, $D^2 = D^{-2}$		

Table 1: Summary of cube move notation

Terminology	Definition or Abbreviation
cubies	The small cube pieces which make up the whole cube.
cubicles	The spaces occupied by the cubies.
facets	The faces of a cubie.
types of cubies: **corner**, **edge**, and **centre**:	A corner cubie has three facets. An edge cubie has two facets. A centre cubie has one facet
home location - of a cubie	The cubicle to which a cubie should be restored.
home position - of a cubie	The orientation in the home location to which a cubie should be restored.
positional names for cube faces	Up (u) Down (d) Right (r) Left (l) Front (f) Back (b)
Notation for cubicles - shown in *italics*	Lower case initials. For example, *uf* denotes the Up-Front edge cubicle
Notation for cubies - shown in *italics*	Upper case initials. For example, *URF* denotes cubie whose home position is in the the Up-Right-Front corner

Table 2: Summary of terminology and notation

Since the center facets are fixed by the basic moves there are only $54-6=48$ facets that move. If we label the facets as in Figure 13 then we see that each basic move corresponds to a rearrangement, or permutation of the numbers 1 through 48. In this way we see that the Rubik's cube is much like the puzzle Swap, in the sense that we have a set of 48 numbers and a set of legal moves M (the 6 basic cube moves) which allow us to rearrange these numbers in some way.

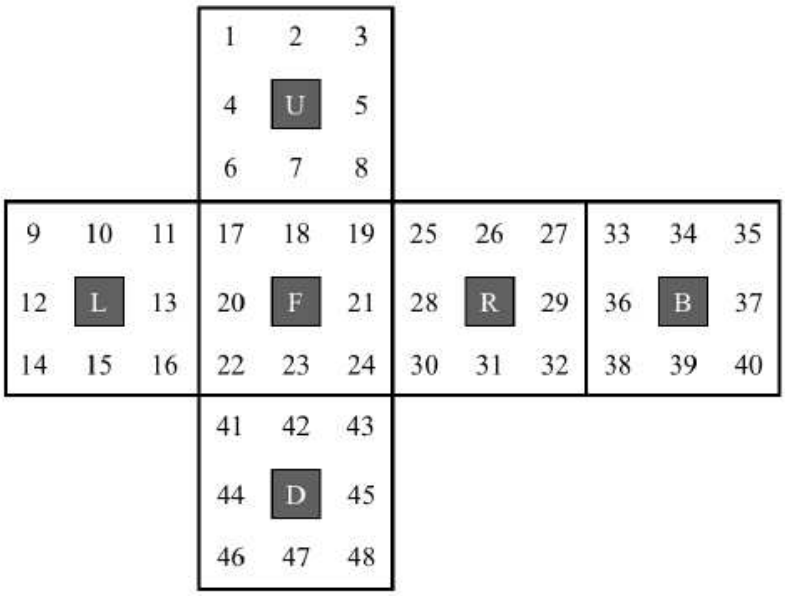


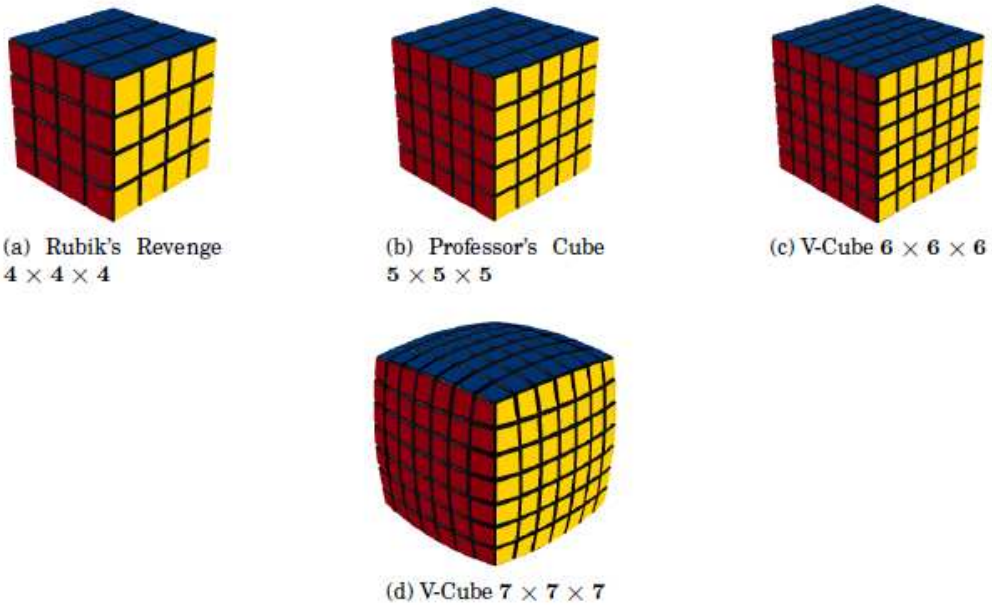
Figure 13. Facet labeling on the $3\times3\times3$ Rubik's cube.

Variations of Rubik's Cube: The Rubik's Cube is the puzzle that started the whole *Twisty Puzzle* craze. Since its invention hundreds of different types of twisty puzzle of all shapes and sizes have been designed. Puzzles of this type are often called *Rubik's Cube-like puzzles*, or *Twisty Puzzles*, or *Permutation Puzzles*. Here we will just briefly describe a few other Rubik's Cube-like puzzles.

$2\times2\times2$ Rubik's cube: The Pocket Rubik's Cube is a cube which is built from smaller cubes where there are 2 cubes along an edge, i.e. a $2 \times 2 \times 2$ cube. The 4 pieces on each face can be rotated, which rearranges the small cubes at that face. The six sides of the puzzle are coloured, so every piece shows three colours. There are no face centres (or centre cubies), and there are no edge cubies, unlike its $3\times3\times3$ counterpart.

$n \times n \times n$ Rubik's cube: Still sticking with the overall shape of a cube, one can increase the number of smaller cubes along each edge. Having 4 smaller cubes along and edge results in a $4\times4\times4$ cube called *Rubik's Revenge*, having 5 smaller cubes along an edge gives a $5\times5\times5$ cube called The Professors Cube (Figure 14).

Being able to create larger versions of the cube turned out to be a substantial design problem, since the resulting object must twist smoothly and be robust enough that it doesn't fall apart easily. Greek inventor Panagiotis Verdes came up with a design that allowed for production of a physical $6\times6\times6$ cube and a $7\times7\times7$ cube [11]. These versions of the cubes are known as *V-Cubes* after the inventor (Figure 14).



(a) Rubik's Revenge $4 \times 4 \times 4$

(b) Professor's Cube $5 \times 5 \times 5$

(c) V-Cube $6 \times 6 \times 6$

(d) V-Cube $7 \times 7 \times 7$

Figure 14. $n\times n\times n$ Cubes

2.2 The Definition of a Permutation Puzzle

The 15-Puzzle, the Oval Track puzzle, Hungarian Rings and Rubik's Cube are all variations of the same theme. Each one consisted of pieces that were rearranged, or permuted, in some way, and the goal is to try to restore the pieces to their original positions. The legal moves that one is allowed to use is forced by the design or construction of the puzzle. Puzzles of this type known as *permutation puzzles* [3].

A **one person game** is a sequence of moves following certain rules which satisfy:

- there are finitely many moves at each stage,
- there is a finite sequence of moves which yields a solution,
- there are no "chance" or "random" moves (such as rolling a dice to determine what to do next),
- there is complete information about each move,
- each move depends only on the present position, not on the existence or non-existence of a certain previous move (such as chess, where castling is made illegal if the king has been moved previously).

A **permutation puzzle** is a one person game (solitaire) with a finite set $T = \{1, 2, ..., n\}$ of puzzle pieces satisfying the following four properties:

1. For some $n > 1$ depending only on the puzzle's construction, each move of the puzzle corresponds to a unique permutation of the numbers in T.
2. If the permutation of T in (1) corresponds to more than one puzzle move then the two positions reached by those two respective moves must be indistinguishable.
3. Each move, say M, must be "invertible" in the sense that there must exist another move, say M^{-1}, which restores the puzzle to the position it was at before M was performed, In this sense, we must be able to "undo" moves.
4. If M_1 is a move corresponding to a permutation τ_1 of T and if M_2 is a move corresponding to a permutation τ_2 of T then $M_1 \cdot M_2$ (the move M_1 followed by the move M_2) is either
   - not a legal move, or
   - corresponds to the permutation $\tau_1\tau_2$.

3 Set Theory

A Rubik's cube is made up a number of different pieces. There are corner cubies, edge cubies, and center cubies as we show before. Each collection of these pieces forms a set. Part of understanding how these pieces move around will be to understand how the cube moves (F, B, R, L, U, D) act on these sets. In this section we recall some basic terminology and notation from set theory which will form the foundation for our mathematical investigations into puzzles.

3.1 Sets and Subsets

A **set** is a well-defined collection of objects. The objects of the set are called elements, and are said to be members of, or belonging to, the set. By well-defined we mean that for any element we wish to consider, we are able to determine, under some scrutiny, whether or not it is a member of the set.

Typically we will use capital letters, such as $A, B, C, ...$ to represent sets and lower case letters to represent elements. For a set A we write $x \in A$ if x is an element of A, and $y \notin A$ if y is not an element in A.

Sets are usually defined in one of two ways:

a) Listing all of its elements in braces: $A = \{a, b, c, ...\}$. For example $A = \{1, 2, 3, 4, 5\}$ is the set of integers from 1 to 5. Therefore, $3 \in A$, but $6 \notin A$.

b) Using *set-builder* notation: $A = \{x \mid x \text{ has property } P\}$. For example we could define the previous set A as $\{x \mid x \text{ is an integer and } 1 \leqslant x \leqslant 5\}$. The vertical bar " | " is read "such that". The symbols $\{x \mid ...\}$ are read "the set of all x such that...".

Some basic sets of numbers are:

- $\mathbb{Z}$= the set of *integers* $= \{..., -2, -1, 0, 1, 2, 3, ...\}$,
- $\mathbb{N}$ = the set of *nonnegative integers* or *natural numbers* $= \{0, 1, 2, 3, ...\} = x \in \mathbb{Z} \mid x \geq 0$,
- $\mathbb{Z}^+$ = the set of *positive integers* $= \{1, 2, 3, ...\} = x \in \mathbb{Z} \mid x > 0$,
- $\mathbb{Q}$ = the set of *rational numbers* $= \left\{\frac{a}{b} \mid a, b \in \mathbb{Z}, b \neq 0\right\}$,
- $\mathbb{Q}^+$ = the set of *positive rational numbers* $= \{x \in \mathbb{Q} \mid x > 0\}$,
- $\mathbb{R}$ is the set of *real numbers*,
- $\mathbb{Z}_n = \{1, 2, ..., n\}$ = the set of integers from 1 to n, where $n \in \mathbb{Z}^+$.

Let A and B be sets. If all the elements of A also belong to B then we say A is a **subset** of B and we write $A \subset B$. For example, $\mathbb{Z}^+ \subset \mathbb{Z}$ since every positive integer is itself an integer, but $\mathbb{Q} \not\subset \mathbb{Z}$, since there are rational numbers that are not integers, consider $\frac{1}{2}$.

Two sets A and B are said to be **equal**, and we write $A = B$, if $A \subset B$ and $B \subset A$.

If a set has a finite number of elements then we say it is a **finite set**. Otherwise it is an **infinite set**. For any finite set A, $|A|$ denotes the number of elements in A and is called the **cardinality**, or size, of A. For example, $|\mathbb{Z}_n| = n$, whereas $\mathbb{Z}$ in an infinite set.

The **empty set**, or **null set**, is the set that contains no elements. The empty set is denoted by $\emptyset$, or $\{\}$, and has that property that $|\emptyset| = 0$.

Let A and B be two sets. The set of all elements belonging to either A or B is denoted $A \cup B$ and is called the **union** of A and B. The set of all elements belonging to both A and B is denoted $A \cap B$ and is called the **intersection** of A and B. The set of all elements not belonging to A is denoted A^c or sometimes by $\bar{A}$, and is called the **complement** of A. The set of all elements in A that are not in B is denoted $A - B$ and is called the **difference** of A with B. We sometimes refer to this as A take away B.

The **Cartesian product** of A and B is the set of all ordered pairs (x, y) where $x \in A$ and $y \in B$ and is denoted by $A \times B$.

The following summarizes the different operations we have on sets:

- $A \cup B = \{x \mid x \in A \textbf{ or } x \in B\}$
- $A \cap B = \{x \mid x \in A \textbf{ and } x \in B\}$

- $A^c = \bar{A} = \{x \mid x \notin A\}$
- $A - B = \{x \mid x \in A \textbf{ and } x \notin B\} = A \cap B^c$
- $A \times B = \{(x, y) \mid x \in A \textbf{ and } y \in B\}$

We call two sets **disjoint** if they have not element in common: A and B are disjoint if $A \cap B = \emptyset$.

3.2 Laws of Set Theory

Some of the major laws that govern set theory are the following.

For any sets $A, B,$ and C taken from a universe U

1. $(A^c)^c = A$ — Law of Double Negation
2. $(A \cup B)^c = A^c \cap B^c$, $(A \cap B)^c = A^c \cap B^c$ — DeMorgan's Laws
3. $A \cup B = B \cup A$, $A \cap B = B \cap A$ — Commutative Laws
4. $A \cup (B \cup C) = (A \cup B) \cup C$, $A \cap (B \cap C) = (A \cap B) \cap C$ — Associative Laws
5. $A \cup (B \cap C) = (A \cup B) \cap (B \cup C)$, $A \cap (B \cup C) = (A \cap B) \cup (B \cap C)$ — Distributive Laws
6. $A \cup A = A$, $A \cap A = A$ — Idempotent Laws
7. $A \cup \emptyset = A$, $A \cap \mathcal{U} = A$ — Identity Laws
8. $A \cup A^c = \mathcal{U}$, $A \cap A^c = \emptyset$ — Inverse Laws
9. $A \cup \mathcal{U} = A$, $A \cap \emptyset = \emptyset$ — Domination Laws
10. $A \cup (A \cap B) = A$, $A \cap (A \cup B) = A$ — Absorbtion Laws

These set theoretic laws are similar to the arithmetic properties of the real numbers, where "$\cup$" plays the role of "$+$", and "$\cap$" plays the role of "$\times$". However, there are several differences.

The next theorem state a result about the cardinality of a disjoint union of sets.

Theorem 3.1. *Let $A_1, A_2, ..., A_n$ be disjoint finite sets. Then*

$$|A_1 \cup ... \cup A_n| = |A_1| + ... + |A_n|.$$

3.3 Examples of Sets Using GAP

Example 3.1. *In this example we show how to define a set, and compute cardinalities, unions, intersections, differences and cartesian products with GAP.*

```
gap> S1:=Set([1,2,3,4,5]);
[ 1, 2, 3, 4, 5 ]
gap> IsSet(S1);
true
gap> S2:=Set([3,4,5,6,7]);
[ 3, 4, 5, 6, 7 ]
gap> S1;S2;
[ 1, 2, 3, 4, 5 ]
[ 3, 4, 5, 6, 7 ]
gap> Size(S1);
5
gap> Size(S2);
5
gap> Union(S1,S2);
[ 1, 2, 3, 4, 5, 6, 7 ]
gap> Intersection(S1,S2);
[ 3, 4, 5 ]
gap> Difference(S1,S2);
[ 1, 2 ]
gap> Difference(S2,S1);
[ 6, 7 ]
gap> Cartesian(S1,S2);
[ [ 1, 3 ], [ 1, 4 ], [ 1, 5 ], [ 1, 6 ], [ 1, 7 ], [ 2, 3 ], [ 2, 4 ],
  [ 2, 5 ], [ 2, 6 ], [ 2, 7 ], [ 3, 3 ], [ 3, 4 ], [ 3, 5 ], [ 3, 6 ],
  [ 3, 7 ], [ 4, 3 ], [ 4, 4 ], [ 4, 5 ], [ 4, 6 ], [ 4, 7 ], [ 5, 3 ],
  [ 5, 4 ], [ 5, 5 ], [ 5, 6 ], [ 5, 7 ] ]
gap> Size(Cartesian(S1,S2));
25
gap> 2 in S1;
true
gap> 1 in S2;
false
```

Example 3.2. *GAP has a number of commonly used sets already built in for integers, positive integers, rationals, positve rationals.*

```
gap> IsInt(1);
true
gap> IsInt(3/5);
false
gap> IsPosInt(8);
true
gap> IsPosInt(-10);
false
gap> IsPosInt(0);
false
gap> IsNegInt(-10);
true
gap> IsRat(1/2);
true
gap> IsPosRat(-1/7);
false
```

Example 3.3. *We can build a set by using properties in GAP. Here we use RemInt(a,b) function which returns the remainder of a when divided by b.*

```
gap> s:=[];;
     for x in [0..10] do
         if RemInt(x,2)=0 then Add(s,x); fi;
     od; s;
[ 0, 2, 4, 6, 8, 10 ]
gap> s:=[];;
     for x in [0..10] do
         if RemInt(x,2)=1 then Add(s,x); fi;
     od; s;
[ 1, 3, 5, 7, 9 ]
```

Example 3.4. *The IsPrimeInt() function returns True if the input is a prime integer, and False if not. Such functions are called boolean valued functions. We can use boolean valued function to create subsets as this example shows.*

```
gap> IsPrimeInt(29);
true
gap> IsPrimeInt(4);
false
gap> s:=[];;
     for x in [0..10] do
         if IsPrimeInt(x) then Add(s,x); fi;
     od; s;
```

```
[ 2, 3, 5, 7 ]
gap> s:=[];;
     for x in [0..1000] do
         if IsPrimeInt(x) then Add(s,x); fi;
     od; Size(s);
168
```

Alternatively, we could use the Filtered(list or coll, func) function in GAP, which returns a new list that contains those elements of the list or collection, respectively, for which the unary function func returns true.

```
gap> Filtered([1..20], IsPrime);
[ 2, 3, 5, 7, 11, 13, 17, 19 ]
```

4 Permutations

The puzzles we have encountered so far all have a common theme: the pieces can be mixed up, and the goal is to restore the pieces back to some proper order. In this section we will introduce some terminology and notation about rearrangements of objects. In particular, we give the definition of a permutation, which is the main object we will use to study puzzles and we will also introduce permutation multiplication, inverses, and order.

In mathematics, the notion of permutation is used with several slightly different meanings, all related to the act of permuting (rearranging in an ordered fashion) objects or values. Informally, a permutation of a set of objects is an arrangement of those objects into a particular order.

Example 4.1. *In the game of Swap on 5 objects the empty puzzle board is shown in Figure 15.*

(a) The empty Swap board

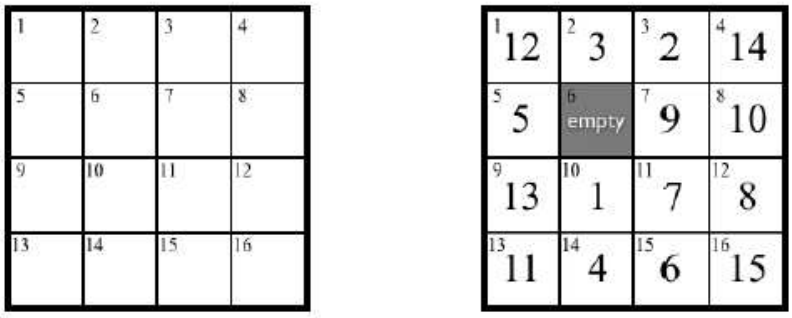
(b) A random arrangement of Swap.

Figure 15. Game of Swap

The puzzle board is filled by laying out the tiles numbered 1 through 5 in the boxes. For example, one such puzzle position is shown in Figure 15b. Each puzzle position corresponds to a permutation of the set $\mathbb{Z}_5 = (1,2,3,4,5)$. *There are 5! = 120 permutations of* $\mathbb{Z}_5$, *and so there are 120 different possible positions in the game of Swap. Only one of which is the solved state.*

```
gap> Factorial(5);
120
```

Example 4.2. *The fifteen puzzle with no tiles in the boxes is shown in Figure 16a. The puzzle is started by placing the 15 tiles anywhere on the board. For example, one such puzzle position is shown in Figure 16b. This corresponds to a permutation of the set* $\mathbb{Z}_{16}$*, where we imagine the blank space as being the* 16^{th} *tile. There are* $16!=20,922,789,888,000$ *permutations of* $\mathbb{Z}_{16}$*, and so there are 16! different ways to lay the tiles on the board.*



(a) The empty 15-Puzzle board

(b) A random arrangement of the 15-Puzzle.

Figure 16. The 15-Puzzle

```
gap> Factorial(16);
20922789888000
```

Example 4.3. *There are six permutations of the objects in the set (1,2,3), namely [1,2,3], [1,3,2], [2,1,3], [2,3,1], [3,1,2] and [3,2,1]. Round brackets (,) denote sets, where the order that elements are listed doesn't matter, however, square brackets [,] denote lists, where the order that elements appear does matter. So as sets (1,2,3) = (2,1,3) but as lists [1,2,3]* $\neq$ *[2,1,3]. We can use GAP to generate permutations of a list, for example [1,2,3].*

```
gap> Set([1,2,3])=Set([2,1,3]);
true
gap> [1,2,3]=[2,1,3];
false

gap> PermutationsList([1,2,3]);
[ [ 1, 2, 3 ], [ 1, 3, 2 ], [ 2, 1, 3 ], [ 2, 3, 1 ], [ 3, 1, 2 ], [ 3, 2, 1 ] ]
gap> NrPermutationsList([1,2,3]);
6
gap> Factorial(3);
6
```

Example 4.4. *Two permutations of the multi-set [a,a,b,b,b] are [b,a,b,a,b] and [b,b,a,a,b]. There are* $\frac{5!}{2!\cdot 3!}$ *= 10 permutations in total, since there are 5! ways to arrange 5 objects, but 2 of the objects are identical, and so are the other 3.*

```
gap> PermutationsList(['a','a','b','b','b']);
[ "aabbb", "ababb", "abbab", "abbba", "baabb", "babab", "babba", "bbaab",
"bbaba", "bbbaa"]
gap> NrPermutationsList(['a','a','b','b','b']);
```

```
10
gap> Factorial(5)/(Factorial(2)*Factorial(3));
10
```

4.1 Definition of a Permutation

Before we give the definition of a permutation we start by recalling the notion of a function, and the properties: one-to-one, and onto.

Definition 4.1. *A **function**, or mapping, f from a (nonempty) set A to a (nonempty) set B is a rule that associates each element $a \in A$ to exactly one element $b \in B$.*

We write $f: A \to B$ to denote a function named f from set A to set B. A is called the **domain** of f and B the **codomain**. If f sends a to b then we write $f(a)=b$, or $f: a \longmapsto b$. We also say b is the image of a under f. The subset of B consisting of all images $f(a)$, for $a \in A$, is called the **range** of f, and is written:

$$f(A) = \{ f(a) \mid a \in A \} \subset B$$

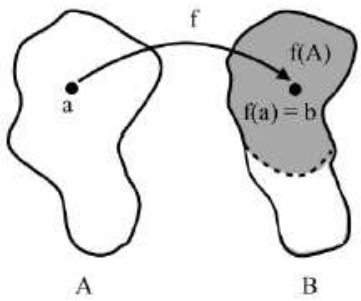


Figure 17. The way to visualize a function, domain, codomain and range.

Definition 4.2. *A function $f: A \to B$ is called **one-to-one**, or **injective**, if each element of B appears at most once as the image of an element of A.*

Definition 4.3. *A function $f: A \to B$ is called **onto**, or **surjective**, if $f(A) = B$. That is, if each element of B is the image of at least one element of A.*

Definition 4.4. *A function that is both injective and surjective is called **bijective**.*

Definition 4.5. *A **permutation** of a set A is a function $a: A \to A$ that is bijective (i.e. both one-to-one and onto).*

Our goal is to understand how the pieces of a puzzle move around, so we typically represent each piece by a number, that is by an element of $\mathbb{Z}_n = \{1, 2, ..., n\}$. A rearrangement of the pieces then corresponds to a bijection from $\mathbb{Z}_n \to \mathbb{Z}_n$, a permutation as defined above.

Unlike in calculus, where most function are defined on infinite sets and given by formulas, permutations of finite sets are usually given by simply listing where each value goes.

For example, we can define a permutation a of the set $\{1,2,3\}$ by stating:

$$a(1)=2, \quad a(2)=1, \quad a(3)=3$$

In GAP we can construct a permutation in the following way:

```
gap> a:=(1,2,3);
(1,2,3)
gap> IsPerm(a);
true
```

A slightly more convenient way to represent this permutation is by:

$$a \leftrightarrow \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}$$

where the top row are the element of $\mathbb{Z}_3$ and the bottom row are the corresponding images under a. This is known as *array notation* for a permutation.

Array Notation: We may define a permutation $a{:}\mathbb{Z}_n \to \mathbb{Z}_n$ by a $2\times n$ array:

$$a \leftrightarrow \begin{pmatrix} 1 & 2 & \ldots & n \\ a(1) & a(2) & \ldots & a(n) \end{pmatrix}$$

Since a is bijective the second row would just be a rearrangement of the numbers in the top row.

The **identity permutation**, denoted by ε or I, is the permutation that does nothing:

$$\varepsilon \leftrightarrow \begin{pmatrix} 1 & 2 & \ldots & n \\ 1 & 2 & \ldots & n \end{pmatrix}.$$

An **n- cycle** is a permutation which cyclically permutes the values. For example,

$$\begin{pmatrix} 1 & 2 & 3 & \ldots & n \\ n & 1 & 2 & \ldots & n-1 \end{pmatrix}.$$

Definition 4.6. *The set of all permutations of the set $\mathbb{Z}_n$ is called the* ***symmetric group*** *of degree n, and is denoted by S_n. in other words,*

$$S_n = \{a : a \text{ is a permutation of } \mathbb{Z}_n\}.$$

Elements of S_n can be written in the form

$$\begin{pmatrix} 1 & 2 & 3 & \ldots & n \\ \alpha(1) & \alpha(2) & \alpha(3) & \ldots & \alpha(n) \end{pmatrix}.$$

It is straightforward to compute the cardinality of the set S_n. There are n choices for $a(1)$. Once $a(1)$ has been chosen, there are $n-1$ possibilities for $a(2)$ (since a is injective we must have $a(1) \neq a(2)$). Once $a(2)$ has been chosen there are $n-2$ choices for $a(3)$. Continuing in this way we see that there are $n \cdot (n-1) \cdot (n-2) \cdots 3 \cdot 3 \cdot 1 = n!$ possible choices for $a(1)$ to $a(n)$. Each choice gives a different permutation. Therefore $|S_n| = n!$.

In GAP the symmetric group, for example S_{10} is defined as follows:

```
gap> SymmetricGroup(10);
Sym( [ 1 .. 10 ] )
```

Another representation of permutations is called *cycle notation*.

Cycle Notation: Consider the 5-cycle permutation a defined as follows:

$$a(1)=2, \quad a(2)=3, \quad a(3)=4, \quad a(4)=5, \quad a(5)=1$$

The array form of a is shown in Figure 18a, and the arrow diagram is shown in Figure 18b. Another arrow diagram which provides a more visual display of the structure of the permutation is shown in Figure 18c. This is called the *cycle-arrow form*.

In this diagram all the information for a is still present. To determine $a(3)$ we look at the diagram and find 3, then we see where the arrow takes it. In this case it takes it to 4, so $a(3)=4$.

The cycle arrow form displays more visually the cycle structure (i.e. we can see the 5 numbers cycling around the circle, which is why we called it a 5-cycle), and it uses only one set of numbered dots, making the diagram more compact than our original arrow form.

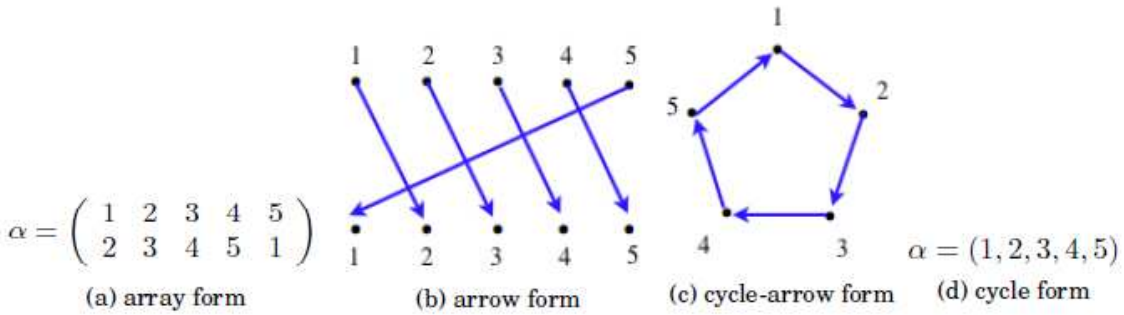


Figure 18. Different representations for a 5-cycle.

However, leaving out the arrows we can simply write the 5-cycle as:

$$a = (1, 2, 3, 4, 5)$$

This represents the fact that a maps each number to the next one in the list, and maps 5 back around to the start of the list, which is 1. This representation is shown in Figure 16d.

All representations in Figure 18 have their own benefits, but it is the cycle notation that is the most compact, and this will be the notation we primarily use in this thesis.

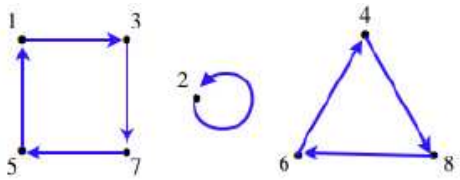
The cycle notation, $a = (1, 2, 3, 4, 5)$, can be read as follows: "1 goes to 2, 2 goes to 3, 3 goes to 4, 4 goes to 5, and 5 goes to 1."

We don't need to start at 1 when writing down the cycle form, if we started at 3, for instance, and constructed the list of numbers we visit by traveling around Figure 18c then we get (3,4,5,1, 2). This is another perfectly acceptable representation of a, reading this cycle notation as described above will tell us exactly how a acts as a function. In particular, we can represent a by any of the equivalent cycle forms:

$$a = (1,2,3,4,5) = (2,3,4,5,1) = (3,4,5,1,2) = (4,5,1,2,3) = (5,1,2,3,4)$$

Despite this notation allowing for non-unique representations of permutations, there is an easy fix. Just writing the cycle so that the first number is the smallest number in the cycle. In this case we would then write $a = (1,2,3,4,5)$ since 1 is the smallest number in this cycle.

Let's look at another permutation: $\beta = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 2 & 7 & 8 & 1 & 4 & 5 & 6 \end{pmatrix}$. The cycle arrow form is:



To construct the cycle form of β we we look at the arrow form above and notice that 1 goes to 3, 3 goes to 7, 7 goes to 5 and 5 goes back to 1. This can simply be written as (1,3,7,5). Similarly, 2 goes to 2 so we write this as (2), and the 4, 6, 8 triangle can be written as (4,8,6). This means we can write β as:

$$\beta = (1,3,7,5)(2)(4,8,6)$$

This is a compact way to represent the permutation β, and we haven't lost any information. For example, we can use the cycle form determine $\beta(3)$ by noticing in (1,3,7,5)(2)(4,8,6) the number 3 is followed by 7, so $\beta(3) = 7$. Similarly, $\beta(5) = 1$ since from 5 we wrap around in the cycle and get back to 1.

If we make one further convention, to leave off any number that gets mapped to itself, then β can be written in a further compact form:

$$\beta = (1,3,7,5)(4,8,6)$$

In this convention, any number not present in the cycle form is assumed to map back to itself.

An expression of the form $(a_1, a_2, ..., a_m)$ is called an m- cycle.

We say β is the product of a 3-cycle and a 4-cycle.

Example 4.5. *To determine the cycle form of the permutation*

$$a = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 \\ 5 & 1 & 6 & 8 & 4 & 10 & 7 & 2 & 9 & 3 \end{pmatrix}$$

we start with the smallest number in the set, in this case it is 1. Since $a(1)=5$ *we begin the cycle by writing* $(1,5,...)...$ *Next,* 5 *maps to* 4*, so we continue building the cycle* $(1,5,4,...)...$

Continuing in this way we construct $(1,5,4,8,2,...)...$ *and since* 2 *maps back to* 1 *then we close off the cycle:*

$$(1,5,4,8,2)$$

Next, we pick the smallest number that doesn't appear in any previously constructed cycle. This is the number 3 *in this case. We now repeat what we just did and construct the cycle involving* 3:

$$(1,5,4,8,2)(3,6,10)...$$

We now pick the smallest number that doesn't appear in any previously constructed cycle, which is 7*, and construct the cycle to which it belongs. In this case* 7 *just maps to itself:*

$$(1,5,4,8,2)(3,6,10)\,(7)...$$

Finally, the only number remaining is 9 *and it maps back to itself so the cycle form of* a *is*

$$(1,5,4,8,2)(3,6,10)\,(7)(9)$$

which simplifies to $a=(1,5,4,8,2)(3,6,10)$*, since our convention is omit* 1*-cycles. Therefore,* a *is the product of a* 3*-cycle and a* 5*-cycle.*

4.2 Permutation Composition

This will be precisely the tool we will need in order to understand how two puzzle moves combine together to give a third.

Let a and β be two permutations of $\mathbb{Z}_n$. We wish to define a new function $a\circ\beta\colon\mathbb{Z}_n\to\mathbb{Z}_n$, called the permutation composition. In order to define a function on $\mathbb{Z}_n$ we just need to specify how it maps the elements. For $k\in\mathbb{Z}_n$ we will define $(a\circ\beta)(k)$ to be the result of first applying a, then applying β to the result. In other words,

$$(a\circ\beta)(k)=\beta(a(k)),\text{for } k\in\mathbb{Z}_n$$

This new function is again a permutation. To see why we just need to observe that it is a bijection.

Injective: Suppose $(a\circ\beta)(k)=(a\circ\beta)(l)$ for some $k,l\in\mathbb{Z}_n$, then $\beta(a(k))=\beta(a(l))$ implies $a(k)=a(l)$, since β is one-to-one. It follows that $k=l$ since a is one-to-one. Therefore, $a\circ\beta$ is one-to-one.

Surjective: Consider any $m\in\mathbb{Z}_n$. Let $l\in\mathbb{Z}_n$ such that $\beta(l)=m$, and let $k\in\mathbb{Z}_n$ such that $a(k)=l$. Both l and k exist since a and β are onto. It follows that $(a\circ\beta)(k)=\beta(a(k))=m$. Therefore, $a\circ\beta$ is onto. This verifies that $a\circ\beta$ is a permutation.

Next we will also drop the symbol $\circ$ to simplify writing.

Definition 4.7. *Let a, $\beta\colon \mathbb{Z}_n \to \mathbb{Z}_n$ be two permutations. The* ***permutation composition****, or* ***product****, of a and β is denoted by $a\beta\colon \mathbb{Z}_n \to \mathbb{Z}_n$ is the permutation defined by:*

$$\begin{array}{rccccc} \alpha\beta: & \mathbb{Z}_n & \to & \mathbb{Z}_n & \to & \mathbb{Z}_n \\ & k & \longmapsto & \alpha(k) & \longmapsto & \beta(\alpha(k)) \end{array}$$

Notice that the composition is opposite to the way functions were combined in calculus.

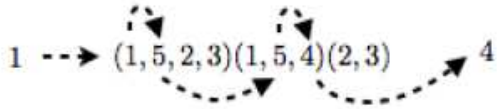
In calculus, and in most branches of mathematics, there is a long standing tradition that variables are to appear to the right of the function: $f(x)$. The composition, $(f \circ g)(x)$ is then read from right-to-left: $f(g(x))$. However, we are defining the composition of permutations as left-to-right. Imagine the move sequence RF^{-1} to a Rubik's cube. The popular convention is to read the move sequence from left-to-right and apply R first, then F^{-1}. This is popular since, for example, this is how you are reading the words on the page right now, from left-to-right. This is precisely the convention we are taking to combine permutations, we combine them from left-to-right.

For example, consider the permutations $a{=}(1,5,2,3)$ and $\beta{=}(1,5,4)(2,3)$ in S_5. What is the cycle form of $a\beta$? Of course, we could just stick the two permutations together, end-to-end, and write

$$a\beta = (1,5,2,3)(1,5,4)(2,3)$$

but it will be more convenient to express the permutation in disjoint cycle form, that is where the various cycles have no numbers in common.

We determine the cycle form of $a\beta$ by determining exactly how it maps each number, beginning with 1. Keep in mind that permutation composition is done from left-to-right, and each cycle that does not contain a number fixes that number. We have that: $(1,5,2,3)$ sends 1 to 5, $(1,5,4)$ sends 5 to 4, and $(2,3)$ fixes 4. So the effect of $a\beta$ is it sends 1 to 4.



Thus we begin writing the disjoint cycle form as $a\beta{=}\ (1,4,\ldots)\ldots$

Repeating this process with 4, we have, cycle-by-cycle, left-to-right,

$$4 \xrightarrow{(1,5,2,3)} 4 \xrightarrow{(1,5,4)} 1 \xrightarrow{(2,3)} 1,$$

so that $a\beta(4) = 1$, and the cycle form is now $a\beta{=}\ (1,4)\ldots$ Next we pick the smallest number that is not in any previously constructed cycle, this would be 2. Repeating this process with 2, cycle-by-cycle, left-to-right,

$$2 \xrightarrow{(1,5,2,3)} 3 \xrightarrow{(1,5,4)} 3 \xrightarrow{(2,3)} 2,$$

so that $a\beta(2) = 2$, and the cycle form is now $a\beta{=}\ (1,4)(2)\ldots$

Continuing in this way we find that $a\beta{=}(1,4)(2)(3,5){=}\ (1,4)(3,5)$.

The important thing when multiplying cycles is to keep moving from one cycle to the next from left-to-right.

In GAP we can compose two permutations by the following way:

```
gap> a:=(1,5,2,3);
```

```
(1,5,2,3)
gap> b:=(1,5,4)(2,3);
(1,5,4)(2,3)
gap> a*b;
(1,4)(3,5)
```

Example 4.6. *Let* $a=(1,4,6,3,7)(2,8)$ *and* $\beta=(2,5,3)(4,7,8,1)$ *be permutations in* S_8. *Then we start with* 1 *under* $a\beta$:

$$1 \xrightarrow{(1,4,6,3,7)} 4 \xrightarrow{(2,8)} 4 \xrightarrow{(2,5,3)} 4 \xrightarrow{(4,7,8,1)} 7,$$

so that $a\beta(1)=7$. *Continuing in this way we find that:*

$$a\beta = (1,4,6,3,7)(2,8)(2,5,3)(4,7,8,1) = (1,7,4,6,2)(3,8,5)$$

and

$$\beta a = (2,5,3)(4,7,8,1)(1,4,6,3,7)(2,8) = (1,6,3,8,4)(2,5,7)$$

In GAP we have:

```
gap> a:=(1,4,6,3,7)(2,8);
(1,4,6,3,7)(2,8)
gap> b:=(2,5,3)(4,7,8,1);
(2,5,3)(4,7,8,1)
gap> a*b;
(1,7,4,6,2)(3,8,5)
gap> b*a;
(1,6,3,8,4)(2,5,7)
```

4.3 Properties of Permutations

a) **Permutation composition is associative**.

This means we don't have to use group brackets when writing long chains of products. The reason that it is associative is simply because permutations are functions, and function composition is associative. To see this, consider permutations a, β, γ: $\mathbb{Z}_n \to \mathbb{Z}_n$. For any $k \in \mathbb{Z}_n$,

$$((\alpha\beta)\gamma)(k) = \gamma(\alpha\beta(k)) = \gamma(\beta(\alpha(k)))$$

and

$$(\alpha(\beta\gamma))(k) = (\beta\gamma)(\alpha(k)) = \gamma(\beta(\alpha(k)))$$

which are the same. So $(\alpha\beta)\gamma = \alpha(\beta\gamma)$.

b) **Every permutation can be written as a product of disjoint cycles.**

This property was implicit before on how to construct the cycle form of a permutation. In particular, when we finished constructing a cycle, the first thing we did was look for a number that did not appear in a previously constructed cycles. This guarantees that our cycles will be disjoint.

c) **Disjoint cycles commute.**

This is also fairly straightforward consequence of the disjoint cycle notation. For example, consider the disjoint cycles $a = (1, 3, 2)$ and $\beta = (4, 5)$. When multiplying these cycles it doesn't matter which order the product is taken: $a\beta = (1,3,2)(4,5) = (4,5)(1,3,2) = \beta a$. Both of these products represent the same permutation: $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 1 & 2 & 5 & 4 \end{pmatrix}$.

Theorem 4.1. (Disjoint Permutations Commute) *If we have two permutations a, β: $\mathbb{Z}_n \to \mathbb{Z}_n$ and have no numbers in $\mathbb{Z}_n$ that are moved by both a and β then $a\beta = \beta a$. In other words, if the disjoint cycle form of a has no number in common with the disjoint cycle form of β then a and β commute.*

As a more physical example of disjoint cycles commuting, consider the moves R and L of Rubik's cube. These moves are disjoint in the sense that there is no common piece that is moved by both R and L. Notice that RL and LR result in exactly the same position of the cube, so RL = LR, and so R and L commute.

d) **Cancellation property**

Lemma 4.1. If permutations $a, \beta, \gamma: \mathbb{Z}_n \to \mathbb{Z}_n$ where $a\beta = a\gamma$ then $\beta = \gamma$.

Similarly, if $\beta a = \gamma a$ then $\beta = \gamma$.

Proof: Multiplying both sides of $a\beta = a\gamma$ on the left by a^{-1} we get

$$a^{-1}(a\beta) = a^{-1}(a\gamma)$$

By assosiativity we have

$$(a^{-1}a)\beta = (a^{-1}a)\gamma$$

and so

$$\varepsilon\beta = \varepsilon\gamma$$

which means $\beta = \gamma$.

A similar argument shows the right cancellation property as well. □

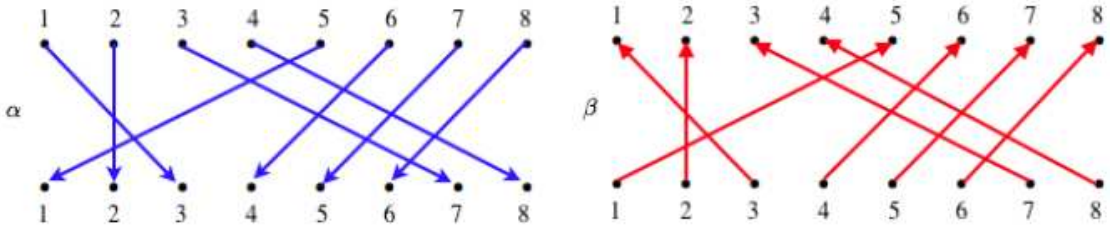
As a consequence of the cancellation property the identity permutation is the only permutation that when multiplied to another permutation it leaves it unchanged. That is, it has the property that $a\varepsilon = a$ for any $a \in \mathbb{Z}_n$. To see this, suppose β is a permutation with this property too, that is $a\beta = a$ for some a. Then $a\beta = a\varepsilon$ and by cancellation of a we have $\beta = \varepsilon$.

4.4 Inverses of Permutations

We will call permutations with the property that their product is the identity, *inverses*. Let's consider two permuations a and β as follows:

$$a = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 2 & 7 & 8 & 1 & 4 & 5 & 6 \end{pmatrix}, \beta = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 5 & 2 & 1 & 6 & 7 & 8 & 3 & 4 \end{pmatrix}$$

We can represent a by an arrow diagram. Each blue arrow represents the mapping defined by the permutation a. If we replace each blue arrow with a red arrow pointing in the opposite direction then we get an arrow diagram representing β (follow arrows from bottom row to top row). In this sense, the inverse permutation is obtained by "reversing the arrows".



We can do the same experiment with the array form of β. Let's flip β over, that is, we'll switch the top and bottom rows:

$$\begin{pmatrix} 5 & 2 & 1 & 6 & 7 & 8 & 3 & 4 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{pmatrix},$$

then let's put the top row in increasing order, while keeping all the columns in tact:

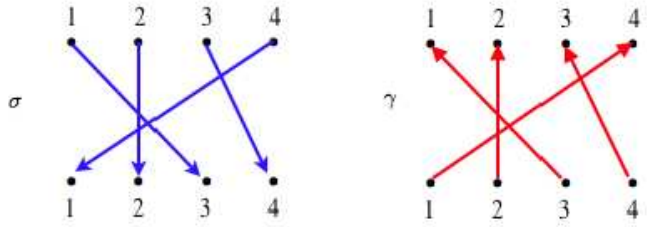
$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 3 & 2 & 7 & 8 & 1 & 4 & 5 & 6 \end{pmatrix}.$$

This is precisely a! Let's recall that the notation means the number in the top row maps to the number directly beneath it in the bottom row. For instance, a maps 1 to 3. If β is to be the inverse of a then it must undo what a does. In particular, it must map 3 back to 1. This means 3 must appear above 1 in the array from of β.

The same is true for every number. In general, we have if k is above m in a (i.e. $a(k)=m$) then m is above k in β (i.e. $\beta(m)=k$). This explains exactly what we observed when we flipped β.

Now suppose, we start with a permutation, say $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 2 & 4 & 1 \end{pmatrix}$ and we flip the rows, and reorder the first row so it is increasing order, while keeping the columns in tact: $\gamma = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 4 & 2 & 1 & 3 \end{pmatrix}$. We see that each number from 1 to 4 appears in the second row, so it is surjective, and no number appears more than once, so it is injective. Therefore, it is a permutation. And by the observation above, it is the inverse of σ, that is, $\sigma\gamma = \varepsilon$.

We can also use the arrow diagram to see this visually, γ was constructed by "reversing the arrows" of σ, so clearly γ is a bijection, and it is the inverse of σ, since it just undoes what σ is doing.



From these observations we obtain that every permutation has an inverse, and it is unique. Moreover, we have a straighforward way to construct inverses given a permutation in array or arrow form.

Theorem 4.2. *For any permutation* $a:\mathbb{Z}_n\to\mathbb{Z}_n$*, there exists a unique permutation* $\beta:\mathbb{Z}_n\to\mathbb{Z}_n$ *such* $a\beta=\beta a=\varepsilon$.

Proof: Let a be a permutation, define a new function $\beta:\mathbb{Z}_n\to\mathbb{Z}_n$ as follows:

$$\beta(m)=k\Leftrightarrow a(k)=m$$

for $k,m\in\mathbb{Z}_n$. Since a is bijective, for any m such a k exists and is unique. It follows that $(a\beta)(m)=a(\beta(m))=a(k)=m$ and $(\beta a)(k)=\beta(a(k))=\beta(m)=k$. This proves the theorem. □

Definition 4.8. *For any permutation* a *the unique permutation* β *such that* $a\beta=\beta a=\varepsilon$ *is called the inverse of* a *and is denoted by* a^{-1}.

Theorem 4.3. *For two permutations* a *and* β,

$$(a\beta)^{-1}=\beta^{-1}a^{-1}$$

In general, the ***inverse of a product permutations*** *is the product of the inverses in the reverse order:*

$$(a_1a_2\ldots a_k)^{-1}=a_k^{-1}\ldots a_2^{-1}a_1^{-1}.$$

Proof: Taking the product, and using associativity of permutation multiplication,

$$(a\beta)^{-1}(\beta^{-1}a^{-1})=a\beta^{-1}\beta^{-1}a^{-1}=a\varepsilon a^{-1}=aa^{-1}=\varepsilon$$

Therefore, $\beta^{-1}a^{-1}$ is the inverse of $a\beta$. A similar argument proves the general statement. □

Every permutation can be written as a product of disjoint cycles: $a=\sigma_1\sigma_2\ldots\sigma_k$. We have already seen that the inverse of a product is the product of the inverses in the reverse order, so

$$a^{-1}=\sigma_k^{-1}\ldots\sigma_2^{-1}\sigma_1{}^{-1}.$$

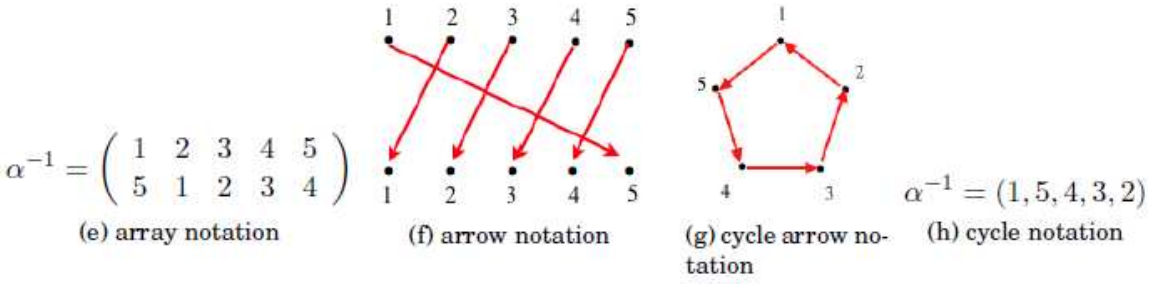
This means, in order to determine a^{-1} directly from its cycle form we just need to know how to find the inverse of a cycle.

Consider the 5-cycle $a = (1, 2, 3, 4, 5)$. We would like to come up with a simple method for determining the inverse a^{-1} directly from the cycle form, and without having to change representation to array form, or arrow form.

We already know that if we have a in array form: $a = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 3 & 4 & 5 & 1 \end{pmatrix}$ then it is easy to write down the inverse: $a^{-1} = \begin{pmatrix} 2 & 3 & 4 & 5 & 1 \\ 1 & 2 & 3 & 4 & 5 \end{pmatrix}$. If we express this back in cycle form we have $a^{-1} = (1, 5, 4, 3, 2)$. An alternative way to write this cycle is $(5, 4, 3, 2, 1)$. This gives us a very simple method for computing an inverse of a cycle, we need just to write the cycle backwards.

$$a^{-1} = (1, 2, 3, 4, 5)^{-1} = (5, 4, 3, 2, 1) = (1, 5, 4, 3, 2)$$

The last equality follows from our convention that we start the cycle with the smallest number in the cycle. Figure 18 and Figure 19 show the various representation of a and a^{-1}.



(e) array notation (f) arrow notation (g) cycle arrow notation (h) cycle notation

Figure 19. Different representation of a^{-1}

Example 4.7.

(a) *The inverse of the permutation* $a = (1, 6, 3, 4, 5)$ *is* $a^{-1} = (5, 4, 3, 6, 1) = (1, 5, 4, 3, 6)$.

```
gap> a:=(1,6,3,4,5);
(1,6,3,4,5)
gap> a^-1;
(1,5,4,3,6)
```

(b) *The inverse of a 2-cycle is itself. For example,* $\beta = (1, 2)^{-1} = (2, 1) = (1, 2)$.

```
gap> b:=(1,2);
(1,2)
gap> b^-1;
(1,2);
```

(c) *The inverse of the permutation* $\gamma = (1, 4, 3, 5)(3, 7, 6)(2, 5, 7, 3, 1)(6, 4)(2, 3, 5, 4)(4, 5, 3)$ *is*

$$\gamma^{-1} = [\,(1, 4, 3, 5)(3, 7, 6)(2, 5, 7, 3, 1)(6, 4)(2, 3, 5, 4)(4, 5, 3)\,]^{-1}$$

$$= (4, 5, 3)^{-1}(2, 3, 5, 4)^{-1}(6, 4)^{-1}(2, 5, 7, 3, 1)^{-1}(3, 7, 6)^{-1}(1, 4, 3, 5)^{-1}$$

$$= (4, 3, 5)(2, 4, 5, 3)(6, 4)(2, 1, 3, 7, 5)(3, 6, 7)(1, 5, 3, 4).$$

Since γ^{-1} *is not in disjoint cycle form (due to the fact that* γ *itself was not), then we should probably put it in this form.*

$$\gamma^{-1} = (1, 6)(2, 7, 3, 4, 5).$$

In GAP, since γ in disjoint form is γ=$(1,6)(2,5,4,3,7)$, we will have:

```
gap> c:=(1,6)(2,5,4,3,7);
(1,6)(2,5,4,3,7)
gap> c^-1;
(1,6)(2,7,3,4,5)
```

4.5 Exponents of Permutations

When we describe moves on Rubik's cube we write things such as: RB^2R^{-1}. Exponents represent inverse moves, such as R^{-1} is the inverse of R, and also they represent repetition of moves, such as B^2 is the move B repeated twice.

If we follow the move sequence $RU^{-2}\ B^2\ R^{-1}DU^{-2}$ with the move U then the complete move sequence would be $RU^{-2}\ B^2\ R^{-1}DU^{-2}U$. But certainly, $U^{-2}U$ simplifies to U^{-1}, since a counterclockwise half turn (U^{-2}) followed by a clockwise quarter turn (U) is equivalent to a counterclockwise quarter turn. This means the complete move sequence is equivalent to $RU^{-2}B^2R^{-1}DU^{-1}$. We write this as $(RU^{-2}B^2R^{-1}DU^{-2})U = RU^{-2}B^2R^{-1}DU^{-1}$.

This notation translates nicely to composition of permutations.

If $a \in S_n$ and m is a positive integer then a^m denotes the product of a with itself m-times. That is $a^m = aa\cdots a$.

We define negative exponents by the rule a^{-m} = $(a^{-1})^m$, where m is any positive integer.

We define the zero exponent by $a^0 = \varepsilon$, where ε is the identity permutation.

An important observation is that some of the rules of the exponents apply to the composition of permutations. Specifically, for any two integers m and k and for any $a \in S_n$, we have

(a) $a^m a^k = a^{m+k}$

(b) $(a^m)^k = a^{\mathrm{mk}}$

However, one property from multiplication of real numbers (ab)^m $= a^m b^m$, is not true for permutations: if $a, \beta \in S_n$ and $m \in \mathbb{Z}$ then in general $(a\beta)^m$ is not equal to $a^m\beta^m$.

For real numbers this property relies on the fact that multiplication of real numbers is commutative. We have already seen this is not the case for permutations under composition.

However, we have the following result.

Lemma 4.2. *If $a, \beta \in S_n$ commute with each other, that is $a\beta$=βa, then for all integers m, $(a\beta)^m$ =$a^m\beta^m$.*

4.6 Order of Permutations

The order of a permutation $a \in \mathbb{Z}_n$ is the smallest positive integer m such that $a^m = \varepsilon$.

Theorem 4.4. *For any $a \in S_n$ there exists a positive number m for which $a^m = \varepsilon$. (The smallest such m is the **order** of a, denoted* ord(a)*.)*

Proof: Consider the set of all powers of $a, \{a^k : k \in \mathbb{Z}^+\}$. Since this is a subset of the finite set S_n it must also be finite. This means all the powers of a cannot be distinct, so there must be k, l such that $a^k = a^l$ where $k > l > 0$. Now multiplying a^{-l} to the left of both sides (i.e. cancelling a^l) we get:

$$a^{-l}a^k = a^{-l}a^l$$

and so

$$a^{k-l} = \varepsilon$$

□

We can now describe precisely which integers m have the property that $a^m = \varepsilon$.

Theorem 4.5 *Let a be a permutation. If $a^m = \varepsilon$ then $\mathrm{ord}(a)$ divides m.*

Proof: Let $n = \mathrm{ord}(a)$, and suppose $a^m = \varepsilon$. By the division algorithm there exist integers q and $0 \leq r < m$ such that $m = \mathrm{qn} + r$. In other words, n goes into m q-times, with r left over. Therefore

$$\varepsilon = a^m = a^{\mathrm{qn}+r} = (a^n)^q a^r = \varepsilon \mathrm{q} a^r = a^r$$

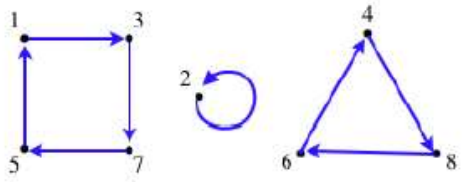
Since r is smaller than the order of a this is only possible if $r = 0$. Hence n divides m. □

To determine the order of a given permutation in array form we have to continue computing powers until we hit the identity. This is a very inefficient way to compute orders. The disjoint cycle form has the enormous advantage of allowing us to "eyeball" the order of a permutation.

For example the 5-cycle $(1, 2, 3, 4, 5)$ has order 5. In general, an m-cycle has order m. The order of a product of disjoint cycles is given by the next theorem.

Theorem 4.6.(Order of a Permutation) *The order of a permutation written in disjoint cycle form is the least common multiple of the lengths of the cycles.*

Before we prove this theorem lets consider the permutation $\beta = (1, 3, 7, 5)(4, 8, 6)$, which is the product of a cycle of length 3 and a cycle of length 4. The arrow diagram is as follows.



We want to determine the smallest power k so that β^k is the identity. Every application of β moves the numbers around the square (4-cycle) one position, so in order to have numbers return to their original position β must be applied 4, or a multiple of 4, times. This means 4 divides k. Similarly, considering the triangle (3-cycle) β would need to be applied a multiple of 3 times to move numbers back to their original positions. This means 3 divides k. Since we require both 3 and 4 to divide k, and we want k to be as small as possible, this means k is the *least common multiple* of 3 and 4, that is $\mathrm{ord}(\beta) = k = \mathrm{lcm}(3, 4) = 12$.

An easy way to see $\beta^{12} = \varepsilon$ is to do the following:

$$\beta^{12} = [(1,3,7,5)(4,8,6)]^{12} = (1,3,7,5)^{12}(4,8,6)^{12} = [(1,3,7,5)^4]^3[(4,8,6)^3]^4 = \varepsilon^3\varepsilon^4 = \varepsilon.$$

Here we used the fact that an m-cycle has order m, and $(\sigma_1\sigma_2)^k = \sigma_1^k\sigma_2^k$, for disjoint cycles σ_1 and σ_2.

This is precisely the idea that we use to give a general proof of the theorem.

Proof:(Theorem 4.6)

One cycle: As we noted above, a cycle of length m has order m.

Two disjoint cycles: Now suppose a and β are disjoint cycles of lengths a and b. Let k be the least common multiple of a and b, that is, k is the smallest positive integer which is divisible by both a and b. Since a and β commute then $(a\beta)^k = a^k\beta^k = \varepsilon$ (here we used that fact that a divides k implies $a^k = \varepsilon$ and b divides k implies $\beta^k = \varepsilon$). It follows from Theorem 4.5 that the order of $a\beta$, call it t, divides k. We now wish to show $t = k$. From $\varepsilon = (a\beta)^t = a^t\beta^t$ it follows that $a^{-t} = \beta^t$. However, a and β have no symbol in common, and since raising a cycle to a power does not introduce new symbols, a^{-t} and β^t also have no symbol in common. Since $a^{-t} = \beta^t$ and have no commons symbols then they both must be the identity: $a^{-t} = \beta^t = \varepsilon$. If follows from Theorem 4.5 that t is divisible by a and b. This means that $k = \text{lcm}(a, b)$ must also divide t. Therefore $t = k$, as desired.

More than two disjoint cycle: The general case involving more than two cycles is handled in a similar way. □

Example 4.8

(a) *The order of* $a = (1, 3, 4)(2, 5)$ *is* $\text{lcm}(3, 2) = 6$. *Observe that*

$a^6 = [(1, 3, 4)(2, 5)]^6 = (1, 3, 4)^6(2, 5)^6 = \varepsilon$

```
gap> a:=(1,3,4)(2,5);
(1,3,4)(2,5)
gap> Order(a);
6
```

(b) *The permutation* $\beta = (1, 7, 4, 10, 3)(2, 5, 6, 9)(8, 11)$ *has order* $\text{lcm}(5, 4, 2) = 20$.

```
gap> b:=(1,7,4,10,3)(2,5,6,9)(8,11);
(1,7,4,10,3)(2,5,6,9)(8,11)
gap> Order(b);
20
```

5 Representation of Puzzles with Permutations

The puzzles we have encountered so far all have a common theme: the pieces can be mixed up, and the goal is to restore the pieces back to some proper order. In other words, the pieces are permuted. In this section we will see how to represent a puzzle by a permutation.

There are two types of permutations associated with a puzzle:

a) the permutation describing the puzzles current position,

b) the permutation corresponding to a move sequence applied to the puzzle.

Though the second one can really be thought of as a special case of the first, since the permutation we assign to a move sequence is just the one which represents the puzzle position after the move is applied to the solved state.

In this section we will see how to write down these permutations for the standard set of puzzles that are presented. The permutation describing the puzzles position is a composition of the permutations corresponding to the puzzle moves which takes the puzzle from the solved state to that position. That is, multiplying the permutations corresponding to the moves, should give us the permutation of the resulting position.

The way we do this will be the same for all puzzles where both the moving pieces and home positions have been labelled by numbers in $\mathbb{Z}_n$.

Definition 5.1 (Puzzle Position to Permutation) *For a given position (scrambling) of the puzzle, the permutation corresponding to this position is* $a: \mathbb{Z}_n \to \mathbb{Z}_n$ *where*

$$a(i) = j \quad \text{if piece labelled } i \text{ moved to position labelled } j.$$

This permutation describes precisely how the pieces in the home (or solved) state configuration were moved to produce the current configuration.

Definition 5.2 (Puzzle Move to Permutation) *For a given move sequence applied to the puzzle, the permutation corresponding to this move sequence is* $\beta: \mathbb{Z}_n \to \mathbb{Z}_n$ *where*

$$\beta(i) = j \quad \text{if the piece in position labelled } i \text{ moved to position labelled } j.$$

These definitions show how to construct the function a which corresponds to a position/move, but we have to ensure if this function is actually a permutation. That is, we want to observe a is one-to-one and onto. To see this, notice in any scrambling of the pieces no position has more than one piece occupying it, in other words, distinct pieces have gone into distinct positions. This means a is a one-to-one map from $\mathbb{Z}_n$ to $\mathbb{Z}_n$, which is then necessarily onto. This observation suggests why a is indeed a permutation.

Theorem 5.1 (Multiplying Moves) *Let* a *be the permutation corresponding to the current position of the puzzle, and* $\beta_1, \beta_2, \ldots \beta_k$ *be a move sequence applied to the puzzle which results in a final position* γ*. Then*

$$a\beta_1\beta_2 \ldots \beta_k = \gamma$$

To see why this is true, consider any piece of the puzzle, say the piece labelled l. Then, before the move sequence is applied, the piece l starts in position $x_0 = a(l)$. As the moves are applied one-byone the l piece moves to position x_1, then to position x_2, and so on, until it finally ends up in position x_k, where

$$x_1 = \beta_1(x_0) = \beta_1(a(l)) = (a\beta_1)(l)$$

$$x_2 = \beta_2(x_1) = \beta_2((a\beta_1)(l)) = (a\beta_1\beta_2)(l)$$

$$\vdots$$

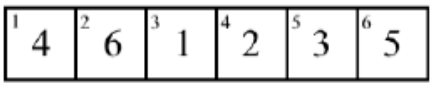
$$x_k = \beta_k(x_{k-1}) = \beta_k((a\beta_1\beta_2 \ldots \beta_{k-1})(l)) = (a\beta_1\beta_2 \ldots \beta_k)(l)$$

Therefore, $\gamma(l)=x_k = (a\beta_1\beta_2...\beta_k)(l)$ for every $l \in \mathbb{Z}_n$, and so $\gamma = a\beta_1\beta_2...\beta_k$. This proves the theorem.

Over the next subsections we will look at each puzzle individually.

5.1 Swap

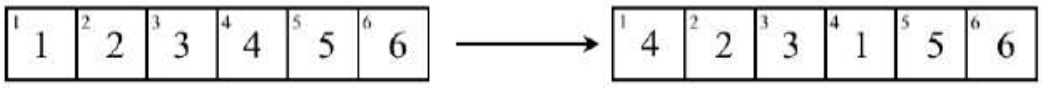
Each arrangement of the numbers in the Swap Puzzle, with n numbers, is a permutation of the set $\mathbb{Z}_n = \{1, 2, 3, ..., n\}$. For example, consider the following position of Swap with 6 numbers.



The permutation $a:\mathbb{Z}_6 \to \mathbb{Z}_6$ we associate to this position is determined as follows. Since tile number 1 moved to box number 3, then $a(1)=3$. Since tile 2 moved to box 4, then $a(2)=4$. Continuing in this way we find a maps numbers 1 through 8 as follows.

$$a = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 3 & 4 & 5 & 1 & 6 & 2 \end{pmatrix} \quad \text{or} \quad \text{in cycle form } a = (1, 3, 5, 6, 2, 4).$$

Consider the move obtained by swapping the tiles in boxes 1 and 4. If we think of applying this move to the solved state of the puzzle, for example:



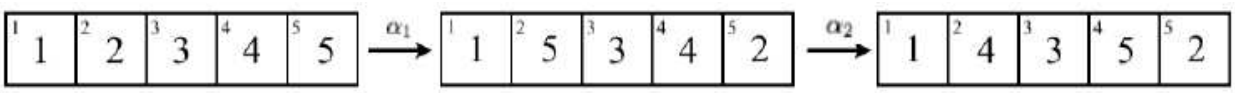
then we can represent this move by the permutation β corresponding to the position it leaves the puzzle in: $\beta=(1, 4)$.

Now imagine the puzzle was not in the solved state, and we were applying the 1, 4 swap. For example, we apply the move as follows:

In order to assign a permutation to this move, the simplest way is to consider it is just the same move as above, ignoring the actually objects in the boxes. All that matters is that the contents of box 1 and box 4 were switched. The permutation should then only depend on the boxes involved and how the contents move between boxes, but it shouldn't depend on what exactly is in the boxes. This is the essence of Definition 5.2.

When we wish to describe a move, we can just state it by giving the corresponding permutation. For example, the permutation (3,7) represents the move of switching the contents of boxes 3 and 7.

Example 5.1 *Consider the following sequence of moves in Swap.*



The first move consists of swapping the contents of boxes 2 *and* 5 *so it corresponds to the permutation* $a_1 = (2, 5)$. *The second move consists of swapping the contents of boxes* 2 *and* 4 *so it corresponds to the permutation* $a_2 = (2, 4)$. *The product* $a_1 a_2 = (2, 5)(2, 4) = (2, 5, 4)$, *which is the permutation representing the move sequence as a whole, is precisely the permutation corresponding to the final position.*

Example 5.2 *Apply the move sequence* $\tau_1 = (3,5), \tau_2 = (1,2), \tau_3 = (2,5), \tau_4 = (1,4)$ *to the game of Swap with* 6 *objects, and draw the final position of the game board, assuming you began with it in the solved state.*

The move sequence corresponds to the single maneuver: $a = \tau_1\tau_2\tau_3\tau_4 = (3, 5)(1, 2)(2, 5)(1, 4)$, *(Theorem 5.1) which means the resulting game board position is as follows.*

1	2	3	4	5	6
4	3	5	2	1	6

We could have also applied the move sequences one-by-one to achieve the same result (here we simply write the numbered tiles, as they appear on the game board, separated by vertical bars |):

$$1|2|3|4|5|6 \xrightarrow{\tau_1=(3,5)} 1|2|5|4|3|6 \xrightarrow{\tau_2=(1,2)} 2|1|5|4|3|6 \xrightarrow{\tau_3=(2,5)} 2|3|5|4|1|6 \xrightarrow{\tau_4=(1,4)} 4|3|5|2|1|6$$

Example 5.3 *Write the permutation* $a = (1, 5, 3, 7)(4, 8, 6)$ *as a product of* 2*-cycles by solving the corresponding Swap puzzle.*

The permutation a corresponds to the position

1	2	3	4	5	6	7	8
7	2	5	6	1	8	3	4

which we will simply write as 7|2|5|6|1|8|3|4. *To solve the puzzle from this state we may do the following:*

$$\alpha \Rightarrow 7|2|5|6|1|8|3|4 \xrightarrow{\tau_1=(1,5)} 1|2|5|6|7|8|3|4 \xrightarrow{\tau_2=(3,7)} 1|2|3|6|7|8|5|4 \xrightarrow{\tau_3=(4,8)} 1|2|3|4|7|8|5|6$$
$$\xrightarrow{\tau_4=(5,7)} 1|2|3|4|5|8|7|6 \xrightarrow{\tau_5=(6,8)} 1|2|3|4|5|6|7|8 \Rightarrow \varepsilon.$$

This means $a\tau_1\tau_2\tau_3\tau_4\tau_5 = \varepsilon$ or $a = \tau_5^{-1}\tau_4^{-1}\tau_3^{-1}\tau_2^{-1}\tau_1^{-1}$. *Therefore, we have found a decomposition of a into* 2*-cycles:*

$$a = (1, 5, 3, 7)(4, 8, 6) = (6, 8)(5, 7)(4, 8)(3, 7)(1, 5).$$

5.2 15-Puzzle

Consider the tiles in the 15 puzzle mixed-up. For example, in Figure 20c each tile was moved from some numbered box (its home box) to some other numbered box: for example the tile in box 1 moved to box 10, but the tile in box 5 stayed in box 5. We represent the empty space as tile number 16, which is also called the "empty tile".

We can write down the permutations describing each of the positions in Figure 20 by using Definition 5.1.

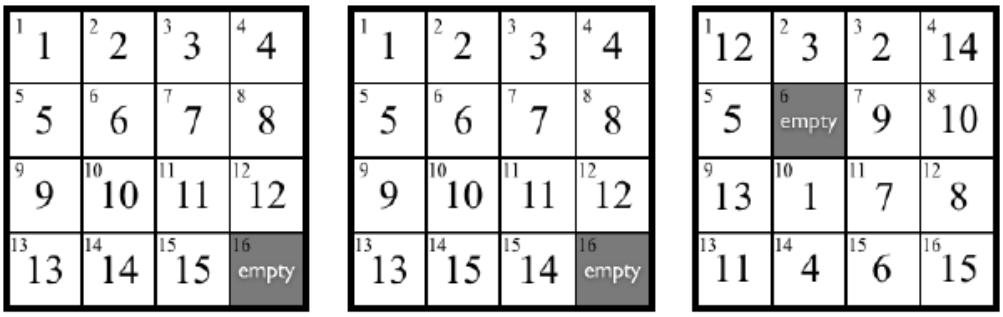


Figure 20. The 15-Puzzle

a) This puzzle is in the solved state, so no tiles have been moved. This corresponds to the identity permutation ε. The array form of this permutation is

$$\varepsilon = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \end{pmatrix}.$$

b) In this puzzle the tiles in boxes 14 and 15 were switched. This corresponds to the permutation $(14, 15)$. The array form of this permutation is

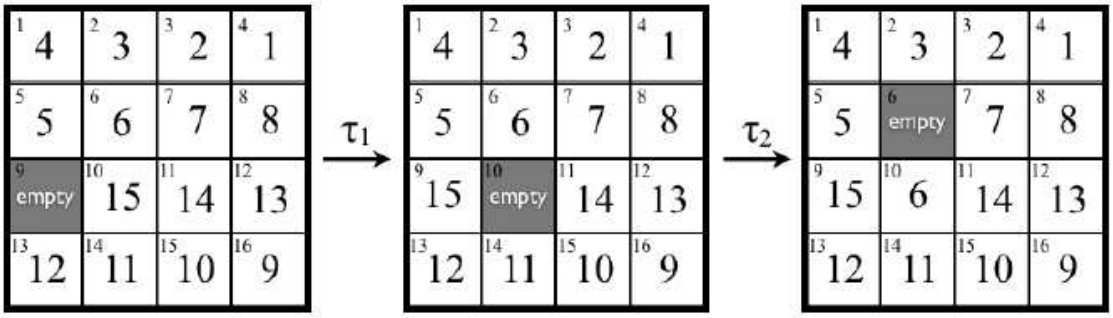
$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 15 & 14 & 16 \end{pmatrix}.$$

c) The tile originally in box 1 (that is, the tile labeled by 1) was moved to box 10, so $1 \longmapsto 10$ for this permutation. The tile originally in box 10 (that is, the labeled by 10) was moved to box 8, so $10 \longmapsto 8$. Continuing in this fashion we construct the cycle form of the corresponding permutation: $(1, 10, 8, 12)(2, 3)(4, 14)(6, 15, 16)(7, 11, 13, 9)$, where we omitted the 1-cycle (5). The array form of this permutation is

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 10 & 3 & 2 & 14 & 5 & 15 & 11 & 12 & 7 & 8 & 13 & 1 & 9 & 4 & 16 & 6 \end{pmatrix}$$

By the construction of the 15-Puzzle a legal move consists of swapping a tile with the empty tile, provided it is adjacent to the empty tile. This means legal moves are 2-cycles, and move sequences are products of 2-cycles.

Example 5.4 *Consider the following sequence of moves in the 15-Puzzle.*



The first move consists of moving the empty space from box 9 *to* 10 *so it corresponds to the permutation* $\tau_1 = (9, 10)$. *The second move consists of moving the empty space form box* 10 *to* 6 *so it corresponds to the permutation* $\tau_2 = (10, 6)$.

The first position is given by $a = (1,\ 4)(2,\ 3)(9,\ 16)(15,\ 10)(11,\ 14)(12,\ 13)$, *the last position is* $\beta = (1, 4)(2, 3)(6, 10, 15, 9, 16)(11, 14)(12, 13)$, *and we have*

$$a\tau_1\tau_2 = (1, 4)(2, 3)(9, 16)(15, 10)(11, 14)(12, 13)(9, 10)(10, 6)$$

$$= (1, 4)(2, 3)(6, 10, 15, 9, 16)(11, 14)(12, 13) = \beta$$

5.3 Oval Track puzzle

Since there are 20 moving disks on the Oval Track puzzle (Figure 21), each position/move can be described as a permutation of $\mathbb{Z}_{20}$.

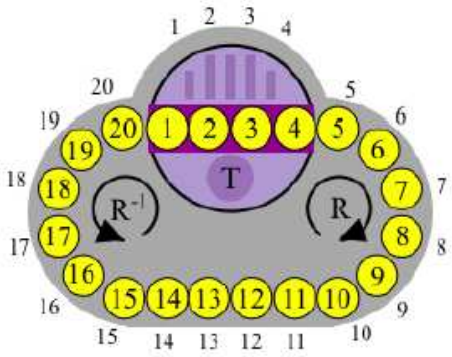


Figure 21. The Oval Track Puzzle

As already mentioned in section 2, the basic legal moves of the oval track puzzle are R, R^{-1} and T, where R denotes a clockwise rotation of numbers around the track, where each number moves one space, R^{-1} denotes a counterclockwise rotation of the numbers around the track, and T denotes a rotation of the turntable (Figure 22).

The permutation corresponding to the legal moves R, R^{-1} and T are as follows:

$$R = (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)$$

$$R^{-1} = (1, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2)$$

$$T = (1, 4)(2, 3)$$

Note that $T^{-1}=T$. This is due to the fact that spinning the turntable in either direction achieves the same result.

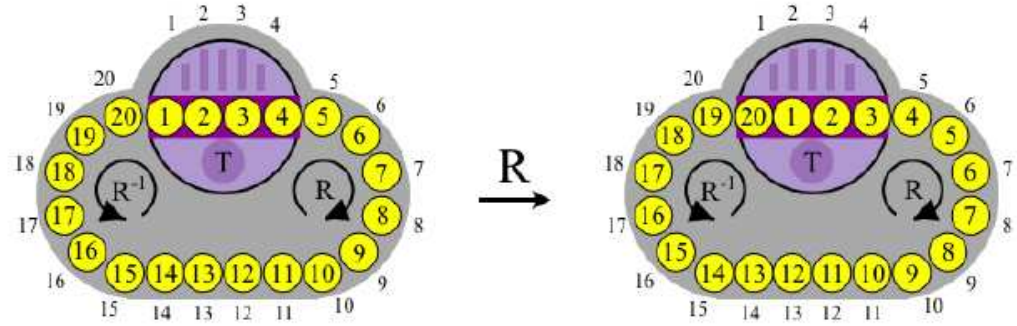


Figure 22. Basic Moves R and T of Oval Track.

Example 5.5 *Express, in cycle form, the permutations describing each of the positions in Figure* 23.

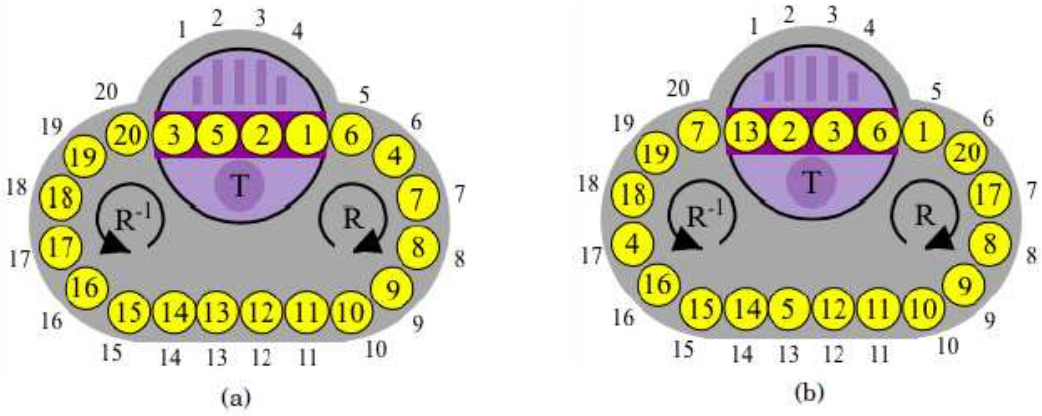


Figure 23. Oval Track scramblings.

a) *Disk* 1 *was moved to slot* 4, *disk* 4 *was moved to slot* 6, *disk* 6 *was moved to slot* 5, *disk* 5 *was moved to slot* 2, *disk* 2 *was moved to slot* 3, *and disk* 3 *was moved to slot* 1. *All other*

disks are still in their home positions, so the corresponding permutation is $(1, 4, 6, 5, 2, 3)$.

b) *Similar to part (a) we just follow where each disk ended up. The corresponding permutation is* $(1, 5, 13)(4, 17, 7, 20, 6)$.

Example 5.6 *For each of the following move sequences, which were applied to the solved-state Oval Track puzzle, draw the resulting configuration of the disks on the puzzle.*

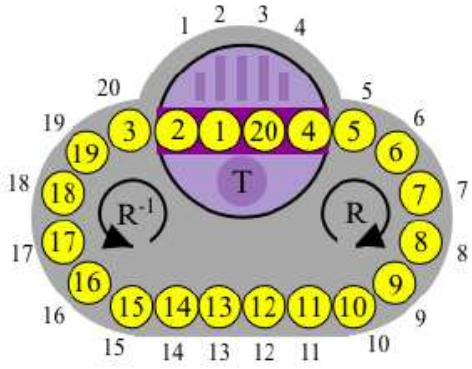
a) RTR^{-1}

b) $R^{-4}TR^2TR^{-1}$

a) *If we have a physical puzzle, or one virtual, then we can actually perform the move sequence and attain the resulting configuration. We can also do this using the permutation representations of the move sequence:*

$$\begin{aligned} RTR^{-1} &= (1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)(1,4)(2,3)R^{-1} \\ &= (1,3)(4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)R^{-1} \\ &= (1,3)(4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20) \\ &\qquad (1,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2) \\ &= (1,2)(3,20) \end{aligned}$$

The resulting position is drawn below.

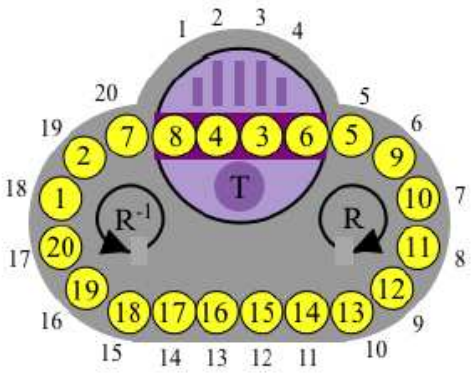


b) *Multiplying two* 20*-cycles and a* 4*-cycle in part (a) was not technically difficult, it was just tedious. This product* $R^{-4}TR^2TR^{-1}$ *would be very tedious, and the actual calculation*

wouldn't be too enlightening. So we will use GAP to do this.

```
gap> R:=(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20);
(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20)
gap> T:=(1,4)(2,3);
(1,4)(2,3)
gap> R^(-4)*T*R^(2)*T*R^(-1);
(1,18,15,12,9,6,4,2,19,16,13,10,7,20,17,14,11,8)
```

$R^{-4}TR^2TR^{-1}$ corresponds to the permutation returned by GAP and the puzzle looks like this



5.4 Hungarian Rings

There are 38 moving disks in the the (numbered) Hungarian Rings puzzle (Figure 24), so each position/move can be described as a permutation of $\mathbb{Z}_{38}$.

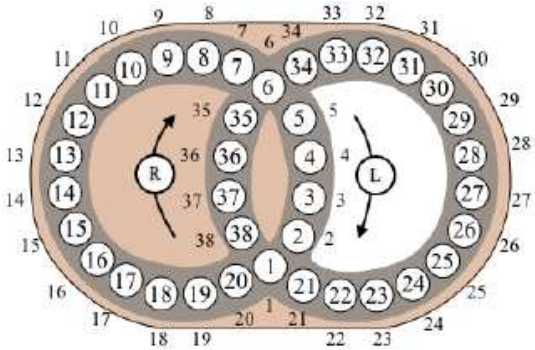


Figure 24. Hungarian Rings - numbered version.

As already mentioned in section 2, the basic legal moves of the Hungarian Rings puzzle are R, R^{-1}, L and L^{-1}, where R denotes a clockwise rotation of numbers around the right-hand ring (each number moves one space), R^{-1} denotes a counterclockwise rotation of the numbers around the right-hand ring, L denotes a clockwise rotation of numbers around the left-hand ring, and L^{-1} denotes a counterclockwise rotation of the numbers around the left-hand ring.

The permutation corresponding to each of the legal moves R and L are:

$$R = (1, 38, 37, 36, 35, 6, 34, 33, 32, 31, 30, 29, 28, 27, 26, 25, 24, 23, 22, 21)$$

$$L = (1, 20, 19, 18, 17, 16, 15, 14, 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2)$$

R^{-1} and L^{-1} correspond to the inverses of these permutations.

Example 5.7 *Express, in cycle form, the permutations describing each of the positions in Figure 25.*

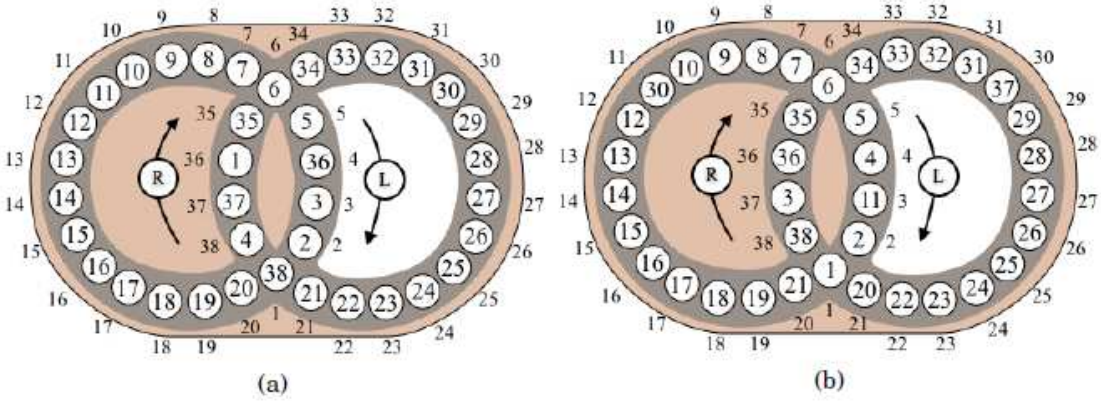


Figure 25. Hungarian Rings scramblings.

a) *We simply follow where each disk has ended up. The corresponding permutation is* $(1, 36, 4, 38)$.

b) *Similar to part (a), following where each disk has ended up, the corresponding permutation is* $(3, 37, 30, 11)(20, 21)$.

Example 5.8 *For each of the following move sequences, which were applied to the solved-state Hungarian Rings puzzle, draw the resulting configuration of the disks on the puzzle.*

a) $R^{-1}LR$

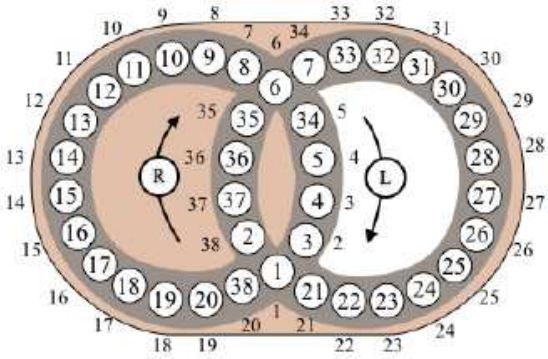
b) $L^5R^5L^{-5}R^{-5}$

a) *If we have a physical puzzle, or one virtual, then we can actually perform the move sequence and attain the resulting configuration. We can also do this using the permutation representations of the move sequence, by multiplying the permutations. We will use GAP to do the computations.*

```
gap> L:=(1,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2);
(1,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2)
gap> R:=(1,38,37,36,35,6,34,33,32,31,30,29,28,27,26,25,24,23,22,21);
```

```
(1,38,37,36,35,6,34,33,32,31,30,29,28,27,26,25,24,23,22,21)
gap> R^(-1)*L*R;
(2,38,20,19,18,17,16,15,14,13,12,11,10,9,8,7,34,5,4,3)
```

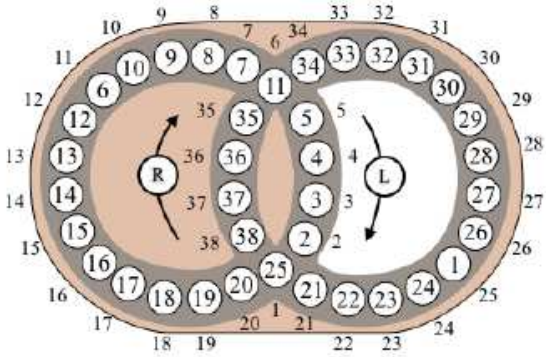
The resulting position is drawn below.



b) *Again, we will use GAP to do the desired calculation.*

```
gap> L^(5)*R^5*L^(-5)*R^(-5);
(1,25)(6,11)
```

The resulting position is drawn below.



5.5 Rubik's Cube

To keep track of how the pieces of the cube move around, and to be able to describe movements and positions by permutations, we label each of the facets with numbers. For the $2\times2\times2$ cube there are 24 facets, whereas for the $3\times3\times3$ cube there are 54, but only 48 actually move. The 6 centres can be thought of as remaining fixed (though they can rotate, but this is only noticeable if the sticker has an image on it).

- **$2\times2\times2$ cube**

We label the facets of the Pocket Cube as shown in Figure 26. Figure 27 shows the labeling on an actual cube.

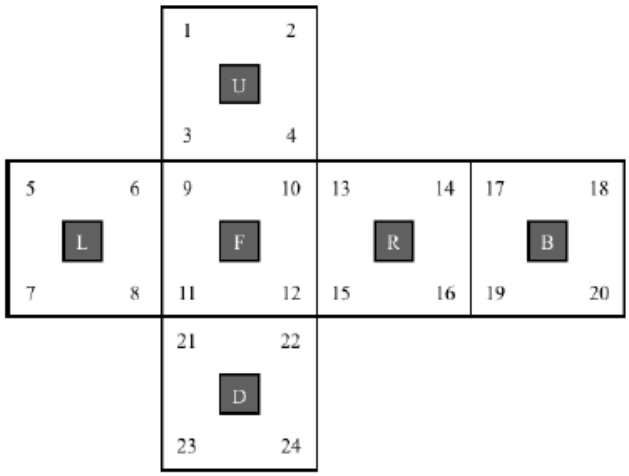
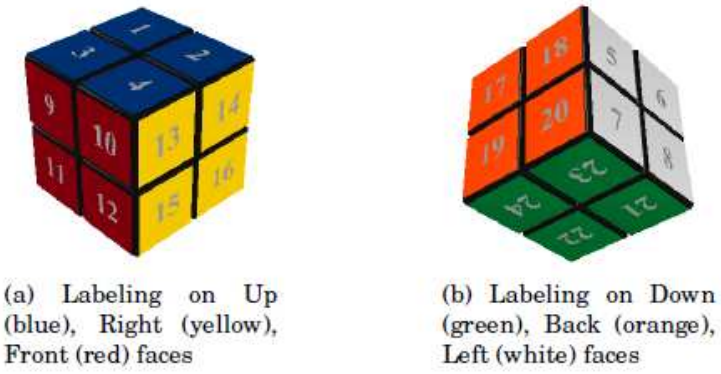


Figure 26. Facet labeling on the Pocket cube.



(a) Labeling on Up (blue), Right (yellow), Front (red) faces

(b) Labeling on Down (green), Back (orange), Left (white) faces

Figure 27. The labeling of the facets of the Pocket Cube.

We associate permutations to positions and moves in the usual way (as described in Definitions 5.1 and 5.2). The basic moves of the Rubik's Cube are R, L, U, D, F, B and their inverses. Each one denotes a clockwise quarter turn of the corresponding face. The permutation corresponding to each of the basic moves of the Pocket Cube are:

$$R = (13, 14, 16, 15)(10, 2, 19, 22)(12, 4, 17, 24)$$

$$L = (5, 6, 8, 7)(3, 11, 23, 18)(1, 9, 21, 20)$$

$$U = (1, 2, 4, 3)(9, 5, 17, 13)(10, 6, 18, 14)$$

$$D = (21, 22, 24, 23)(11, 15, 19, 7)(12, 16, 20, 8)$$

$$F = (9, 10, 12, 11)\,(3, 13, 22, 8)\,(4, 15, 21, 6)$$

$$B = (17, 18, 20, 19)(1, 7, 24, 14)\,(2, 5, 23, 16)$$

$R^{-1}, L^{-1}, U^{-1}, D^{-1}, F^{-1}, B^{-1}$ correspond to the inverses of these permutations.

- **$3 \times 3 \times 3$ cube**

We label the facets of the Rubik's Cube as shown Figure 28. Figure 29 shows the labeling on an actual cube.

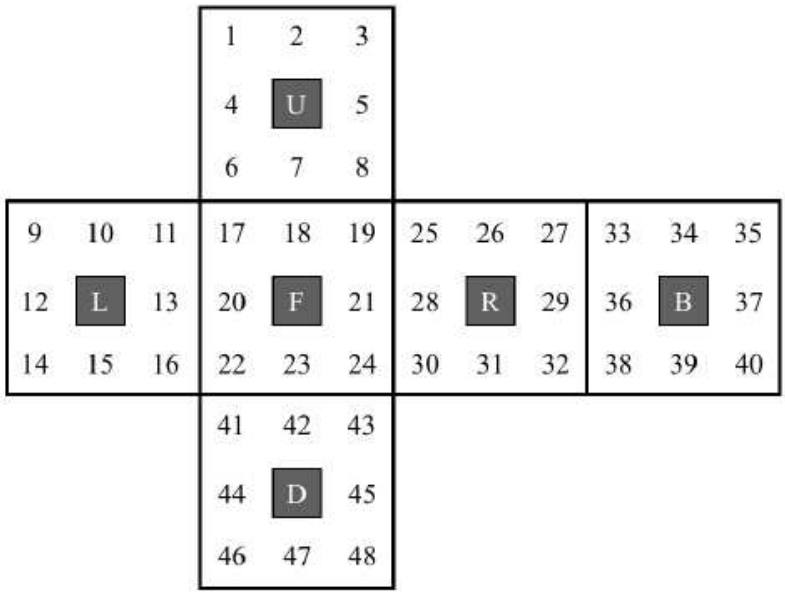
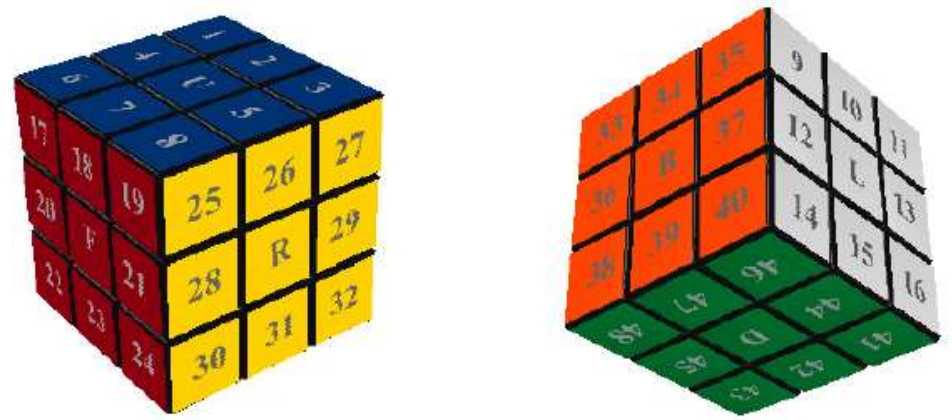


Figure 28. Facet labeling on the Rubik's cube.



(a) Labeling on Up, Right, Front faces

(b) Labeling on Down, Back, Left faces

Figure 29. The labeling of the facets of Rubik's Cube.

The permutation corresponding to each of the basic moves of the Rubik's Cube are [3]:

$$R=(25,27,32,30)(26,29,31,28)(3,38,43,19)(5,36,45,21)(8,33,48,24)$$

$$L=(9,11,16,14)(10,13,15,12)(1,17,41,40)(4,20,44,37)(6,22,46,35)$$

$$U=(1,3,8,6)(2,5,7,4)(9,33,25,17)(10,34,26,18)(11,35,27,19)$$

$$D=(41,43,48,46)(42,45,47,44)(14,22,30,38)(15,23,31,39)(16,24,32,40)$$

$$F=(17,19,24,22)(18,21,23,20)(6,25,43,16)(7,28,42,13)(8,30,41,11)$$

$$B=(33,35,40,38)(34,37,39,36)(3,9,46,32)(2,12,47,29)(1,14,48,27)$$

$R^{-1}, L^{-1}, U^{-1}, D^{-1}, F^{-1}, B^{-1}$ correspond to the inverses of these permutations.

Here is how we can input the permutations for the Rubik's Cube into GAP.

```
gap> R:=(25,27,32,30)(26,29,31,28)(3,38,43,19)(5,36,45,21)(8,33,48,24);
(3,38,43,19)(5,36,45,21)(8,33,48,24)(25,27,32,30)(26,29,31,28)
gap> L:=(9,11,16,14)(10,13,15,12)(1,17,41,40)(4,20,44,37)(6,22,46,35);
(1,17,41,40)(4,20,44,37)(6,22,46,35)(9,11,16,14)(10,13,15,12)
gap> U:=(1,3,8,6)(2,5,7,4)(9,33,25,17)(10,34,26,18)(11,35,27,19);
(1,3,8,6)(2,5,7,4)(9,33,25,17)(10,34,26,18)(11,35,27,19)
gap> D:=(41,43,48,46)(42,45,47,44)(14,22,30,38)(15,23,31,39)(16,24,32,40);
(14,22,30,38)(15,23,31,39)(16,24,32,40)(41,43,48,46)(42,45,47,44)
gap> F:=(17,19,24,22)(18,21,23,20)(6,25,43,16)(7,28,42,13)(8,30,41,11);
(6,25,43,16)(7,28,42,13)(8,30,41,11)(17,19,24,22)(18,21,23,20)
gap> B:=(33,35,40,38)(34,37,39,36)(3,9,46,32)(2,12,47,29)(1,14,48,27);
(1,14,48,27)(2,12,47,29)(3,9,46,32)(33,35,40,38)(34,37,39,36)
```

We could now, for instance, see what the move sequence RU does to the cube.

```
gap>  R*U;
(1,3,38,43,11,35,27,32,30,17,9,33,48,24,6)(2,5,36,45,21,7,4)(8,25,19)
(10,34,26,29,31,28,18)
```

We can easily compute the order of this permutation, it is lcm(15,7,3) = 105.

Also, since RU consists of a 15-cycle, a 3-cycle and two 7-cycles, raising it to the power of 15 would get rid of the 15- and 3-cycles, and would leave us with some 7-cycles. This means we can move fewer pieces by taking powers of some move sequences.

```
gap> (R*U)^15;
(2,5,36,45,21,7,4)(10,34,26,29,31,28,18)
```

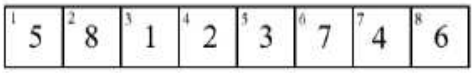
6 Permutations: Products of Transpositions

To solve a permutation puzzle one must determine how the permutation representing the current position of pieces can be decomposed into permutations representing the legal moves. In this section we will focus on this "decomposition problem" and show how every permutation can be decomposed as a product of 2-cycles or 3-cycle [4]. We will also see how this is connected to the solvability of the Swap puzzle and we will also introduce the parity theorem[4]. It is standard terminology to refer to a 2-cycle or 3-cycle as a transposition.

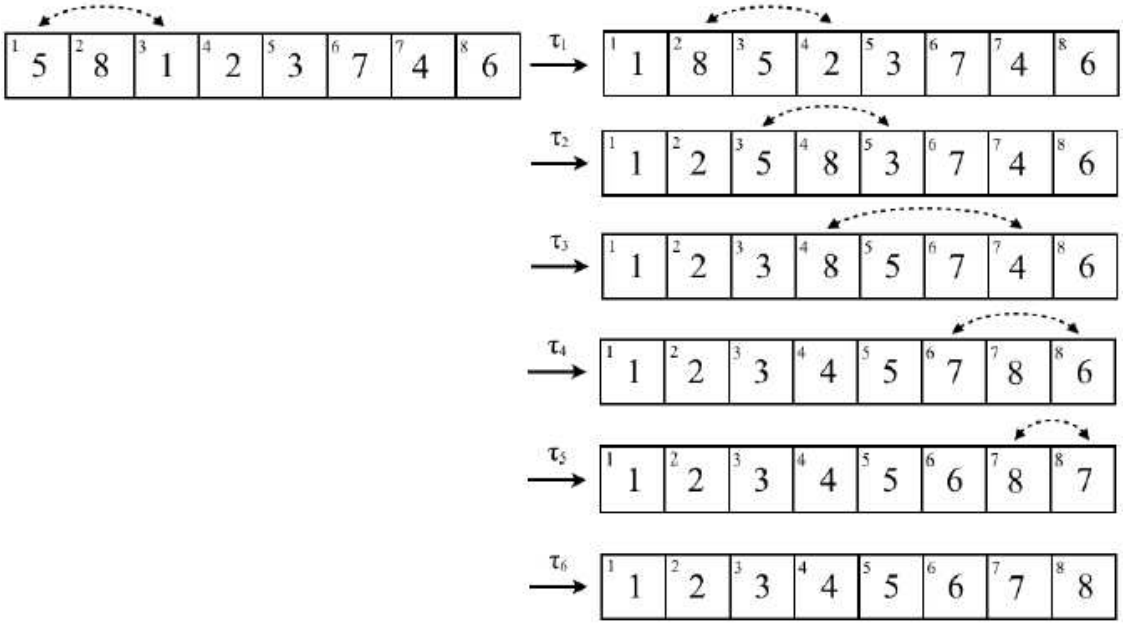
6.1 Products of 2-cycles

Consider the permutation $a = (1, 3, 5)(2, 4, 7, 6, 8)$. We would like to show it can be written as a product of 2-cycles.

To this permutation we consider the corresponding scramble of the Swap puzzle on 8 objects.



To solve the puzzle, the objective is to restore all numbered tiles to their home positions where the only legal moves are to swap tiles from any two boxes (i.e. a 2-cycle). One possible play is as follows.



The dotted arrows indicate the two tiles that are about to be swapped.

The permutations corresponding to the moves are:

$$\tau_1 = (1,3), \quad \tau_2 = (2,4), \quad \tau_3 = (3,5), \quad \tau_4 = (4,7), \quad \tau_5 = (6,8), \quad \tau_6 = (7,8)$$

and so the game-play corresponds to the composition: $a\tau_1\tau_2\tau_3\tau_4\tau_5\tau_6 = \varepsilon$. It follows that:

$$a = \tau_6^{-1}\tau_5^{-1}\tau_4^{-1}\tau_3^{-1}\tau_2^{-1}\tau_1^{-1} \tag{1}$$

$$= (7,8)(6,8)(4,7)(3,5)(2,4)(1,3) \tag{2}$$

This is precisely what we wanted, a is written as a product of 2-cycles.

Our strategy was to just move the numbers, one at a time, to their home positions, and we chose to do this in increasing order, though we could have done it an any order we wanted. This means we should be able to write any permutation as a product of 2-cycles.

Theorem 6.1 (Product of 2-Cycles) *Every permutation in S_n, $n > 1$, can be expressed as a product of 2-cycles.*

Playing with the Swap puzzle showed us intuitively why the theorem is true, it also gave us a method for finding such a decomposition into 2-cycles. As quick as it was to find a decomposition, we will require a much quicker method. Having to draw a Swap game each time we want to compute a decomposition into 2-cycles would be too time consuming.

Decomposition of a *k*-cycle into 2-cycles:

A k-cycle $(a_1, a_2, a_3, ..., a_{k-1}, a_k)(a_1, a_2, a_3, ..., a_{k-1}, a_k)$ in S_n can be decomposed into 2-cycles as follows:

$$(a_1, a_2, a_3, ..., a_{k-1}, a_k) = (a_1, a_2)(a_1, a_3)\cdots(a_1, a_{k-1})(a_1, a_k)$$

Using this method of decomposing k-cycles we can easily decompose any permutation by first writing the permutation as a product of disjoint cycles, and then decomposing each cycle into 2-cycles. For example, consider $a=(1,3,5)(2,4,7,6,8)$. By decomposition we have:

$$a = (1,3,5)(2,4,7,6,8) = (1,3)(1,5)(2,4)(2,7)(2,6)(2,8).$$

Proof:(Theorem 6.1) First note that the identity can be expressed as (1,2)(1,2), and so it is a product of 2-cycles. (This is why we needed $n>1$ in the statement of the theorem.) Now consider any permutation $a \in S_n$. We already know we can write a as a product of disjoint 2-cycles:

$$a = (a_1, a_2, ..., a_r)(b_1, b_2, ..., b_s)\cdots(c_1, c_2, ..., c_t)$$

and each cycle can be decomposed into 2-cycles as we observed above:

$$a = (a_1, a_2)(a_1, a_3)\cdots(a_1, a_r)(b_1, b_2)(b_1, b_3)\cdots(b_1, b_s)\cdots(c_1, c_2)(c_1, c_3)\cdots(c_1, c_t)$$

□

6.2 The Parity Theorem

We have already seen that the permutation $a=(1,3,5)(2,4,7,6,8)$ can be written as a product of 2-cycles in two different ways:

$$a = (7,8)(6,8)(4,7)(3,5)(2,4)(1,3)$$

$$a = (1,3)(1,5)(2,4)(2,7)(2,6)(2,8)$$

The first decomposition we obtained by considering the permutation as an initial scrambling of the tiles of Swap, then solving the puzzle by restoring each tiles to its home position in increasing order, beginning with tile 1. The second decomposition was obtained using the method for decomposing permutations. There are many more possible decompositions of a, here are two more:

$$a = (1,6)(6,7)(1,4)(1,7)(2,8)(4,8)(1,5)(3,5)$$

$$= (1,3)(1,2)(1,4)(1,2)(1,5)(1,2)(1,7)(1,6)(1,8)(1,2)$$

The number of 2 cycles used in the decompositions are not always the same. In the four decompositions we have, two use 6, one uses 8, and one uses 10. Even though the number of 2-cycles isn't constant, it always seems to be of the same parity, in this case it is even. We say for an integer m that its parity is even if m is a multiply of 2. If m is not a multiple of 2 then its parity is odd. Much in the same way that integers come in one of two types, based on parity: odd or even, permutations also come in one of two types, based on the parity of a permutation. This is what the next theorem says.

Theorem 6.2 (The Parity Theorem) *If a permutation a can be expressed as a product of an even number of 2-cycles, then every decomposition of a into 2-cycles must have and even number. On the other hand, if a can be expressed as a product of an odd number of 2-cycles, then every decomposition of a into 2-cycles must have an odd number. In symbols, if*

$$a = \tau_1, \tau_2, \ldots, \tau_r = \sigma_1, \sigma_2, \ldots, \sigma_s$$

where the τ_i's and σ_i's are 2-cycles, then r and s are both even or both odd.

The Parity Theorem tells us that we can define what we mean by the parity of a permutation. If we decompose it as a product of 2-cycles in any way, then the parity of the number of 2-cycles that we used is either odd or even, and this is the parity we assign to the permutation. Here is the formal definition.

Definition 6.1 (Even and Odd Permutation) *A permutation that can be expressed as a product of an even number of 2-cycles is called an **even permutation**. A permutation that can be expressed as a product of an odd number of 2-cycles is called an **odd permutation**.*

Definition 6.2 (Sign of a Permutation) *The sign of a permutation a is defined to be 1 if a is even, or -1 if a is odd.*

$$\text{sign}(a) = \left\{ \begin{array}{ll} 1 & \text{if } a \text{ is an even permutation,} \\ -1 & \text{if } a \text{ is an odd permutation.} \end{array} \right\}$$

Parity of the Identity:

The identity permutation ε is an even permutation.

This follows from $\varepsilon = (1,2)(1,2)$, which is a decomposition into even number of 2-cycles.

In GAP we use the function SignPerm(permutation) to define the sign of a permutation:

```
gap> SignPerm(());    #for the identity permutation
1
```

Parity of a Cycle:

An m-cycle, $(a_1, a_2, \ldots, a_m)$ is an even permutation if m is odd, and it is an odd permutation if m is even.

This follows from the fact that an m-cycle can be expressed as a product of $m-1$ transpositions:

$$(a_1, a_2, \ldots, a_m) = (a_1, a_2)(a_1, a_3)\cdots(a_1, a_m)$$

If m is even then $m-1$ is odd, and vice-versa. This is why the parity of the permutation is opposite to the parity of the length of the cycle.

```
gap> SignPerm((1,2,3,4));
```

```
-1
gap> SignPerm((1,2,3,4,5));
1
```

Example 6.1 *Determine whether the following permutations are odd or even.*

a) $(1,5,11,6,7,3)$

b) $(1,4,12)(3,8,5,9)(7,10)$

a) *This is a 6-cycle and therefore an odd permutation since it can be written as a product of 5 transpositions:*

$$(1,5,11,6,7,3)=(1,5)(1,11)(1,6)(1,7)(1,3).$$

```
gap> SignPerm((1,5,11,6,7,3));
-1
```

b) *Writing each cycle as a product of transpositions we have:*

$$(1,4,12)(3,8,5,9)(7,10)=(1,4)(1,12)(3,8)(3,5)(3,9)(7,10).$$

Since $(1,4,12)(3,8,5,9)(7,10)$ *can be written as the product of 6 transpositions, it follows that it is even.*

```
gap> SignPerm((1,4,12)(3,8,5,9)(7,10));
1
```

6.3 Solvability of Swap

A permutation a is obtainable as a puzzle position of Swap if and only if it can be expressed as a product of legal moves (2-cycles):

$$a=\tau_k^{-1}\cdots\tau_2^{-1}\tau_1$$

If a is the current position then the moves required to solve the puzzle are $\tau_1,\tau_2,\ldots,\tau_k$.

Since every permutation is a product of 2-cycles (Theorem 6.1), then as a consequence we have the following:

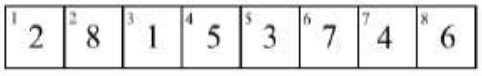
Corollary 6.1 *The Swap puzzle, where the legal moves consist of swapping contents of any two boxes, is solvable from any configuration. In other words, all permutations in* S_n *can be obtained in the Swap puzzle on* n*-objects.*

Notice, the result only applies to Swap when the legal moves are swapping contents of any two boxes. We could consider other variations of Swap, for example:

Variation 1: Legal moves consist of swapping the contents of any other box with the object in box 1. For this variation, a permutation a is obtainable as a position if and only if it can be written as a product of 2-cycles of the form: $(1, a)$ for $a \in \mathbb{Z}_n$.

Variation 2: Legal moves consist of picking any 3 boxes and cycling their contents either to the left or right (i.e. 3-cycles). For this variation, a permutation a is obtainable as a position if and only if it can be written as a product of 3-cycles. Since 3-cycles are even permutations and products of even permutations are even then any product of 3-cycles must be an even permutation. This means an odd permutation of Swap is not solvable under this variation of the legal moves.

For example, the scrambling



is not solvable. This corresponds to the permutation $a = (1, 3, 5, 4, 7, 6, 8, 2)$ which is an odd permutation.

Rather than trying many repeated failed attempts at solving the puzzle, the characterization of a as an odd permutation lead us to abandon any such question to solve the puzzle. This provides a glimpse into how we could use the Parity Theorem to investigate the solvability of puzzles.

6.4 Proof of the Parity Theorem

In this section we provide the proof of the Parity Theorem, however we will prove another result (Claim 6.2), from which the Parity Theorem follows [4].

Claim 6.1 *Any expression for the identity permutation ε as a product of transpositions uses an even number of them. That is, if*

$$\varepsilon = \tau_1 \tau_2 \cdots \tau_m$$

where the τ_i's are transpositions, then m is an even integer.

Suppose $\tau_1 \tau_2 \cdots \tau_r$ and $\sigma_1 \sigma_2 \ldots \sigma_s$ are two decompositions of a permutation a into 2-cycles. Then

$$\varepsilon = a a^{-1} = (\tau_1 \tau_2 \cdots \tau_r)(\sigma_1 \sigma_2 \ldots \sigma_s)^{-1} = \tau_1 \tau_2 \cdots \tau_r \, \sigma_s^{-1} \ldots \sigma_2^{-1} \sigma_1^{-1}$$

is a decomposition of ε into $r+s$ transpositions. If Claim 6.1 is true, then $r+s$ must be even, from which it follows that r and s have the same parity. Therefore the Parity Theorem 6.2 is true.

Therefore, in order to prove the Parity Theorem it is sufficient to prove Claim 6.1.

Claim 6.2 *If there is an expression $\tau_1 \tau_2 \cdots \tau_m$ for the identity permutation ε that uses m transpositions, then there is an expression for ε that uses $m-2$ transpositions.*

Let's assume to the contrary that it was possible to have an expression $\tau_1\tau_2\cdots\tau_m$ for ε where m is odd. Then, assuming Claim 6.2 is true, we could get an expression using $m-2$ transpositions (which is still an odd number of transpositions). We could keep applying Claim 6.2, reducing the number of transpositions by 2 each time, until we end up with an expression for ε using only one transposition. But this is impossible since a single transposition is not equal to the identity (the two numbers in the cycle would not be fixed by the permutation). The fact that we get something impossible from the assumption that an expression for ε exists that uses an odd number of transpositions forces us to conclude that Claim 6.1 is true.

To summarize we have

$$\text{Claim } 6.2 \Rightarrow \text{Claim } 6.1 \Rightarrow \text{Parity Theorem } 6.2$$

So it suffices to prove Claim 6.2.

Proof of Claim 6.2:

First we will pick the right-most occurrence of any number appearing in decomposition into transpositions. Then we will push this number to the left through the transpositions, while transforming the transpositions at the same time, until we eventually get two transpositions that cancel.

Before giving a formal proof we present an example. The product of the following 12 transpositions is the identity.

$$\varepsilon = (1,2)(1,3)(1,4)(1,6)(1,5)(3,4)(3,5)(2,5)(2,3)(1,5)(2,6)(2,4) \tag{3}$$

We will transform this product to a product of only 10 transpositions, which still represents the identity. We choose a number appearing in any transpositions, for example number 3 and we find the right-most transposition containing this number. In this case it would be the transposition $(2,3)$ (the ninth one in the list). We now want to push 3 to the left, so in this product we replace $(2,5)(2,3)$ with the equivalent permutation $(3,5)(2,5)$:

$$\varepsilon = (1,2)(1,3)(1,4)(1,6)(1,5)(3,4)(3,5)\mathbf{(3,5)(2,5)}(1,5)(2,6)(2,4).$$

Now we can replace $(3,5)(3,5)$ with ε:

$$\varepsilon = (1,2)(1,3)(1,4)(1,6)(1,5)(3,4)(2,5)(1,5)(2,6)(2,4)$$

which is an expression using 2 fewer transpositions than we started with.

Sometimes it may take a few more steps, for example if we decided to use 5 instead of 3, we would have proceeded as follows: Find the right-most transposition containing this number in Equation (3). In this case it would be the transposition (1,5). We now want to push 5 to the left, so in this product we replace (2,3)(1,5) with the equivalent permutation (1,5)(2,3), since disjoint cycle commute.

$$\varepsilon = (1,2)(1,3)(1,4)(1,6)(1,5)(3,4)(3,5)(2,5)(1,5)(2,3)(2,6)(2,4)$$

Next we replace $(2,5)(1,5)$ with $(1,5)(1,2)$:

$$\varepsilon = (1,2)(1,3)(1,4)(1,6)(1,5)(3,4)(3,5)(1,5)(1,2)(2,3)(2,6)(2,4),$$

then we replace $(3,5)(1,5)$ with $(1,5)(1,3)$:

$$\varepsilon = (1,2)(1,3)(1,4)(1,6)(1,5)(3,4)(1,5)(1,3)(1,2)(2,3)(2,6)(2,4).$$

Since (3,4) and (1,5) commute, the two (1,5)'s would be canceled and we get:

$$\varepsilon = (1,2)(1,3)(1,4)(1,6)(3,4)(1,3)(1,2)(2,3)(2,6)(2,4).$$

which is an expression using 2 fewer transpositions than we started with.

After these two examples, we now give the formal proof.

Choose a number a that appears in the transposition τ_m. Since $(i,j)=(j,i)$ for any transposition (i,j), the product $\tau_{m-1}\tau_m$ can be expressed in one of the following ways as shown on the left:

$$(a,b)(a,b) = \varepsilon$$

$$(a,c)(a,b) = (a,b)(b,c)$$

$$(c,d)(a,b) = (a,b)(c,d)$$

$$(b,c)(a,b) = (a,c)(c,b)$$

If the first case occurs we may delete $\tau_{m-1}\tau_m$ in the original product and obtain a product for ε using $m-2$ transpositions. In the other three cases we replace the form $\tau_{m-1}\tau_m$ with what appears on the right to obtain a new product of m transpositions that is still the identity, but where the right-most occurrence of a has now moved one 2-cycle to the left. We now repeat the process, where at each stage either we cancel two 2-cycles (and we are done), or we form a new product where a has moved another 2-cycle to the left. This process must terminate with a product of $(m-2)$ transpositions equal to the identity, because otherwise we have a product of m transpositions equal to the identity in which the only occurrence of a is in the left-most 2-cycle, and such a product does not fix a whereas the identity does. □

This completes the proof of Claim 6.2, and therefore the proof of the Parity Theorem too.

6.5 The Alternating Group A_n

As we have already seen there are two types of permutations: even and odd. We will denote the set of all even permutations by A_n, and the set of all odd permutations by O_n [4]. Since every permutation is either odd or even, and no permutation is both, it follows that

$$S_n = A_n \cup O_n, \quad \text{where } A_n \cup O_n = \emptyset.$$

The set A_n of even permutations is closed under composition, closed under taking inverses, and contains the identity. The set O_n of odd permutations is closed under taking inverses, but definitely not closed under composition, nor does it contain the identity. In fact, the composition of any two permutations in O_n is always in A_n.

By A_n (or, in general, any subset of B of S_n) is **closed under composition**, we mean that for any $a, \beta \in A_n$ (or in B) the composition $a\beta \in A_n$ (in B). Similarly, by **closed under taking inverses** we mean that for any $a \in A_n$ (or in B) the inverse permutation a^{-1} is also in A_n (in B).

Let's check why our statements about A_n and O_n are true. The product of any two even permutations is another even permutation so A_n is closed under composition. The identity permutation is even and therefore in A_n. For any permutation $a \in A_n$, it's inverse a^{-1} is also even, since once way to express a^{-1} as a product of transpositions is to just write the ones expressing a in reverse order. So if an even number were used to express a then an even number can be used to express a^{-1}. Similarly, if $\beta \in O_n$ then $\beta^{-1} \in O_n$. The product of two odd permutations gives a permutation that can be expressed in terms of an *odd + odd = even* number of transpositions, and therefore is an even permutation.

In sub-section 4.1 (Definition 4.6), we defined the set of all permutations to be the *Symmetric Group*, S_n. This set has the following four properties regarding composition:

a) Closure. The product of two elements $a, \beta \in S_n$ is another element $a\beta \in S_n$.

b) Associativity. Permutation composition is associative: $a(\beta\gamma)=a\beta(\gamma)$.

c) Identity. The identity (or "do nothing") permutation ε is in S_n. It has the property that $\varepsilon a = a$ for all $a \in S_n$.

d) Inverses. Every $a \in S_n$ has an *inverse* in S_n denoted by a^{-1}. The defining property of an inverse is $aa^{-1} = a^{-1}a = \varepsilon$.

If we look back at all the computations we have done with permutations we see that we are making extensive use of these properties. For example, the cancellation property: $a\beta = a\gamma$ implies $\beta = \gamma$, and $\beta a = \gamma a$ implies $\beta = \gamma$ is a direct consequence of these four properties, as we have already seen at the proof of it in sub-section 4.3. This means that any set of objects, equipped with an operation that combines two to produce a third, and the operation satisfies these four properties, also has the cancellation property. For example, $\mathbb{R}$ under the operation of addition, $+$, satisfies these four properties (identity is 0), so it must also have the cancellation property. The set of invertible 2×2 matrices, under matrix multiplication, satisfies these four properties, so it must also have the cancellation property.

A set A that comes equipped with an operation to combine pairs of elements (add/multiply/compose) such that A is closed under the operation, the operation is associative, there is an identity in A, and inverses exist in A, is called a **group**. Our study into permutations puzzles will essentially consist of considering the set of all legal move sequences, we call this set M, and this set is a subset of S_n which is also a group. (Composition of legal moves is a legal move, composition is associative, there is a "do-nothing" move, and for each move there is a way to "undo" it.) Therefore to each permutation puzzle we can associate a group M of legal move sequences.

What we've shown above is that A_n is a group, whereas O_n is not. O_n fails to contain the identity, nor is it closed under composition.

Definition 6.3 (Alternating Group of Degree n) *The set of even permutations of S_n is denoted by A_n, and is called the alternating group of degree n:*

$$A_n = \{a \in S_n : a \text{ is an even permutation}\}$$

Sometimes A_n could be referred as the set of even permutations.

Theorem 6.3 (Cardinality of A_n) $|A_n| = |O_n| = \frac{n!}{2}$, *for* $n \geq 2$.

Proof: To see this we will pair up all the even permutations a with odd permutations $(1,2)a$, to observe there are equal numbers of each.

If we consider the set of all elements in S_n of the form $(1,2)a$ where $a \in A_n$, and denote this set by $(1,2)A_n$:

$$(1,2)A_n = \{(1,2)a : a \in A_n\}$$

We observe that $(1,2)A_n \subset O_n$, since extending an even permutation by a transposition is an odd permutation. On the other hand, for $\beta \in O_n$ we have $(1,2)\beta \in A_n$ and so $\beta = (1,2)(1,2)\beta \in (1,2)A_n$. Since β was just any element of O_n this means, $O_n \subset (1,2)A_n$. It follows that $O_n = (1,2)A_n$.

Next we note that $(1,2)A_n$ and A_n have exactly the same number of elements. To see this, we just observe that the function $\varphi : A_n \to (1,2)A_n$ defined by $\varphi(a) = (1,2)a$ is a bijection.

Therefore $|A_n| = |O_n|$ and $|A_n| + |O_n| = |S_n|$. Since $|S_n| = n!$ it follows that $A_n = O_n = \frac{n!}{2}$. □

Example 6.2 *List the elements of* A_4.

These are the permutations in S_4 *which are even. The most straightforward way to list elements is to do it in disjoint cycle form, so we'll begin with the identity:*

$$\varepsilon$$

Next, we list elements involving cycles of length at most 2*. And since we want even permutations we don't included single transpositions:*

$$(1,2)(3,4), \quad (1,3)(2,4), \quad (1,4)(2,3)$$

Next we can list 3*-cycles:*

$$(1,2,3), \quad (1,3,2), \quad (1,2,4), \quad (1,4,2), \quad (1,3,4), \quad (1,4,3), \quad (2,3,4), \quad (2,4,3).$$

This is all the elements of A_4*, and there are* 12 *as predicted by Theorem* 6.3*.*

6.6 Products of 3-cycles

Theorem 6.4 *Every permutation in* A_n*, for* $n \geq 3$*, can be expressed as a product of* 3 *cycles.*

Proof: Suppose a is an even permutation, then we can express it as the product of an even number of 2-cycles:

$$a = \tau_1 \tau_2 \cdots \tau_{2k-1} \tau_{2k}$$

We'll group together adjacent pairs of 2-cycles as follows:

$$a = (\tau_1 \tau_2)(\tau_3 \tau_4) \cdots (\tau_{2k-1} \tau_{2k})$$

It suffices to show that a product of two transpositions can either be dropped from the expression or be expressed as a product of 3-cycles, without changing the value of the expression.

Each product $\tau_i\tau_{i+1}$ can be expressed in one of the following ways as shown on the left, depending on whether the transpositions move two things in common, one thing in common, of nothing in common:

$$(a,b)(a,b)=\varepsilon$$

$$(a,b)(a,c)=(a,b,c)$$

$$(a,b)(c,d)=(a,b,c)(a,d,c)$$

If the first case occurs we may delete $\tau_i\tau_{i+1}$ in the original product. In the other two cases we replace $\tau_i\tau_{i+1}$ with what appears on the right to obtain a new product of 3-cycles. □

Example 6.3 *Express the even permutation* $a=(1,6,4)(2,3,7,8)(9,10)$ *as a product of* 3*-cycles.*

To do this the first thing we do is express it as a product of transpositions:

$$a=(1,6)(1,4)(2,3)(2,7)(2,8)(9,10)$$

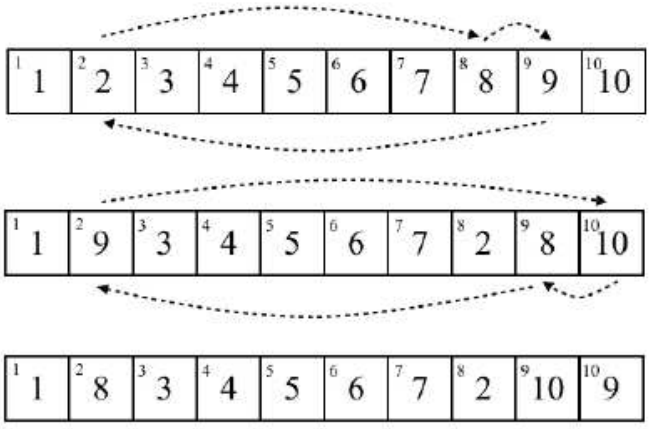
Then we group adjacent transpositions and express each in terms of 3*-cycles.*

$$(1,6)(1,4)=(1,6,4)$$

$$(2,3)(2,7)=(2,3,7)$$

$$(2,8)(9,10)=(2,8,9)(2,10,9)$$

The following simple game of Swap shows how we can express $(2,8)(9,10)$ *as the product of two* 3*-cycles.*



In general, this is precisely the result we used in the proof of the theorem. The way we came up with it there was to look at a simple game of Swap on four objects $a|b|c|d$*. To swap* a *with* b *and* c *with* d *we can first cycle* abc *to the right:* $c|a|b|d$*. Then we can cycle objects in positions* acd *to the left:* $b|a|d|c$*.*

Now we can put everything back together to get:

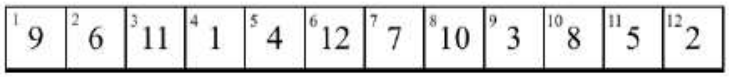
$$a=(1,6,4)(2,3,7)(2,8,9)(2,10,9).$$

6.7 Variation of Swap

We consider the following variation of Swap [4].

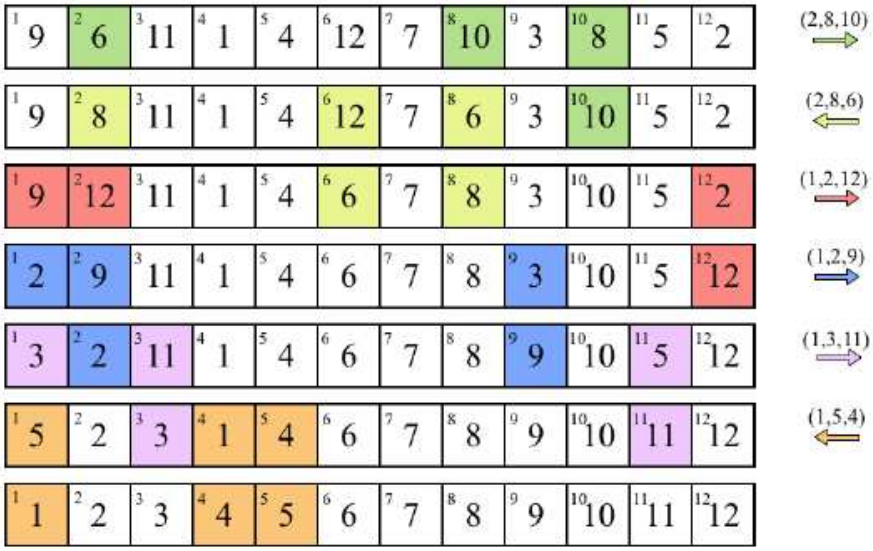
Variation: Legal move is to pick any 3 boxes and cycle their contents either to the left or right.

For example, we suppose the puzzle started in the following position:



The corresponding permutation is $a=(1,4,5,11,3,9)(2,12,6)(8,10)$.

We can solve the puzzle as follows. In each line the shaded boxes represent our choice of 3 boxes, and the arrow on the right indicates which direction the contents are being moved. We also summarize the move by writing the corresponding 3-cycle above the arrow.



In term of permutations this move sequence is:

$$a(2,8,10)(2,8,6)(1,2,12)(1,2,9)(1,3,11)(1,5,4)=\varepsilon$$

or in other words,

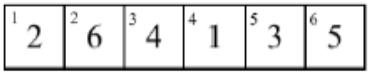
$$a=[(2,8,10)(2,8,6)(1,2,12)(1,2,9)(1,3,11)(1,5,4)]^{-1}$$

$$=(1,4,5)(1,11,3)(1,9,2)(1,12,2)(2,6,8)(2,10,8).$$

That is, considering a as a starting position for this variation of Swap, solving the puzzle is equivalent to expressing a as a product of 3-cycles. Since we know only even permutations are expressible as products of 3-cycles this give us a very simple solvability condition for this variation of Swap.

Corollary 6.2 (Solvability of Swap Variation) *The Swap puzzle, where the legal moves consist of 3-cycles on any three boxes, is solvable only when the starting position is an even permutation. In other words, only even permutations can be obtained in this variation of Swap.*

To see this solvability condition in action, we consider the following scramble of Swap.



If we try to solve it using only 3-cycles, we would very quickly realize it is a difficult task. It is possible to get all but two numbers back into their home positions. In fact, this position corresponds to the permutation $(1, 4, 3, 5, 6, 2)$ which is a 6-cycle, and therefore an odd permutation. Therefore, by Corollary 6.2 no matter how long we play with the puzzle we don't have a hope of solving it.

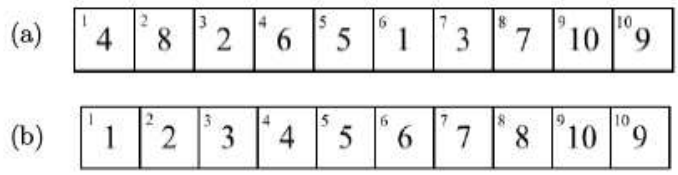


Figure 30. Scrambles of Swap

If we consider the srambles of Swap in Figure 30, we see that the permutation in Figure 30(a) is (1, 6, 4)(2, 3, 7, 8)(9, 10) which is even and hence solvable, whereas the permutation in Figure 30(b) is (9, 10) which is odd, and therefore not solvable. Just knowing Figure 30(a) is solvable doesn't actually solve the puzzle. This is equivalent to expressing (1, 6, 4)$(2, 3, 7, 8)(9, 10)$ as a product of 3-cycles, which we've already done in Example 6.3 there we found $(1, 6, 4)(2, 3, 7, 8)(9, 10) = (1, 6, 4)(2, 3, 7)(2, 8, 9)(2, 10, 9)$. So applying the inverse of this permutation: (2, 9, 10)(2, 9, 8)(2, 7, 3)(1, 4, 6) will solve the puzzle. On the other hand, knowing the puzzle in Figure 30(b) is not solvable means we can abandon playing with it, since there is not way to solve it.

7 The 15-puzzle

We now have enough theory developed to give a full analysis of the 15-puzzle. We will present a solvability criteria which will allow us to easily see whether a given scrambling of the puzzle is solvable. We will also sketch the strategy for solving the puzzle and we will present the solution of the 15-puzzle programmed in GAP.

7.1 Solvability of the 15-puzzle

In order to determine the solvability of a scrambling of the tiles on the 15-puzzle, let's first consider the case where a scrambling places the empty space back into its original box (box 16). This means the corresponding permutation a fixes 16: $a(16) = 16$. We can think of such a permutation as an element of S_{15}. (Considering the disjoint cycle form, 16 does't appear since it is mapped back to itself.)

Figure 31 shows three different configurations of the 15-puzzle corresponding to permutations in S_{15}. The permutations are written below each puzzle. We would like to be able to quickly determine which configurations are solvable.

The next theorem [4] says any rearrangement of tiles in the 15-puzzle starting from the solved-state configuration which brings the empty space back to its original position must be an even permutation of the other 15 pieces. Moreover, it says that every even permutation of the 15 tiles can be obtained as a position on the 15-puzzle.

Theorem 7.1 (Solvability Criteria for 15-Puzzle - Part 1) *A permutation a of the* 15*-puzzle which fixes* 16*, is solvable if and only if it is even: i.e.* $a \in A_{15}$.

It follows that the number of solvable positions of the 15-Puzzle, where the empty space is in its home position, is $|A_{15}| = \frac{15!}{2} = 653,837,184,000$.

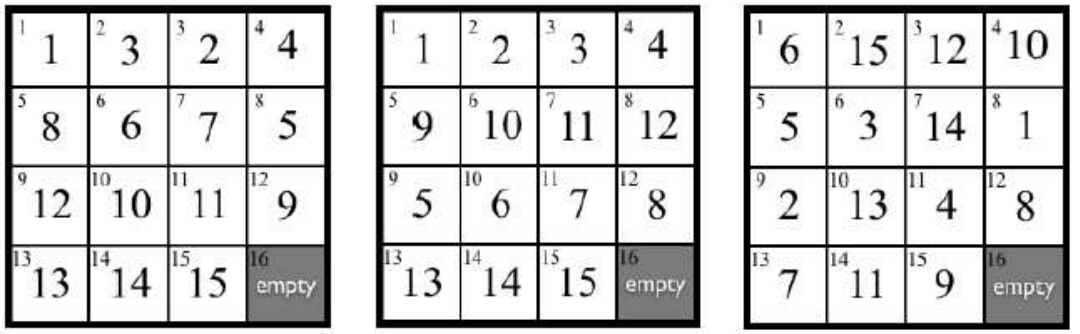


Figure 31. Scrambles of 15-puzzle.

We immediately conclude from this theorem that the puzzles in Figures 31a and 31c are not solvable since the permutations are odd, whereas the puzzle in Figure 31b is solvable since the permutation is even.

In the case when the scrambling does not place the empty space back in box 16, we will see that simply knowing the parity of the permutation and the position of the empty space is enough to determine solvability.

In order to determine the *parity of a box*, we colour the 15-puzzle like a checker board as in Figure 32. We will call the shaded boxes *even* and the white boxes *odd*. Under this definition boxes $1, 3, 6, 8, 9, 11, 14, 16$ are even, whereas the other boxes are odd.

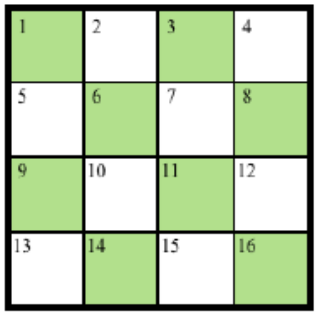


Figure 32. Parity of box: Define the shaded boxes to be even and the white boxes to be odd.

With this concept of odd and even boxes now defined, we can state the general solvability criteria for the 15-puzzle.

Theorem 7.1 (Solvability Criteria for 15-Puzzle - Part 2) *A permutation of the* 15*-puzzle is solvable if and only if the parity of the permutation is the same as the parity of the location of the empty space.*

When the empty space is in one particular box, there are: $|A_{15}| = \frac{15!}{2} = 653,837,184,000$ possible positions of the tiles. Since there are 16 different places to put the empty space, there is a total of $16\left(\frac{15!}{2}\right) = \frac{16!}{2} = 10,461,394,944,000$ possible ways to rearrange the tiles on the board so that the puzzle is solvable. This means, of all 16! ways to arrange the tiles in the boxes, exactly half are solvable.

As an example, the permutation corresponding to the scrambling in Figure 33 is

$$(1,10,11,7,6)(2,3,4,8,12,16,5)(13,15)$$

which is odd and the parity of the location of the empty space is odd, therefore the puzzle is solvable by the solvability criteria: Theorem 7.1-(Part 2).

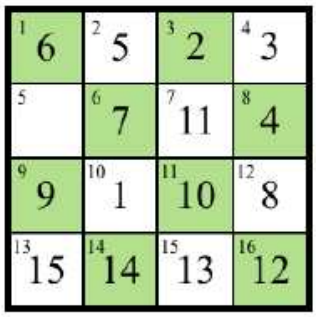


Figure 33. 15-puzzle scrambling.

7.2 Proof of Solvability Criteria

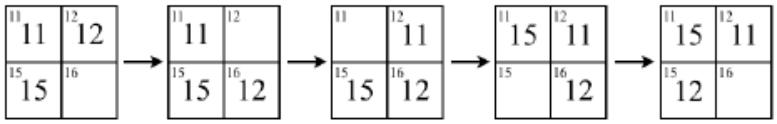
We will prove Theorem 7.1-(Part 1) and then show that Theorem 7.1-(Part 2) is a direct consequence of it [4].

Proof of Theorem 7.1 - Part 1: There are two directions we need to prove: (i) a solvable configuration is an even permutation, and (ii) every even permutation is a solvable configuration. The proof of (i) is very straightforward, but the proof of (ii) requires us to use the fact that even permutations can be expressed using 3-cycles.

(i) Suppose we have a solvable rearrangement of the 15 tiles, where the empty space is in its home position (box 16). Let $a \in S_{15}$ be the corresponding permutation. Since puzzle moves consist of transpositions - the empty space is swapped with an adjacent tile - then let τ_1, τ_2, ..., τ_k be the moves (i.e. transpositions) which solve the puzzle (i.e. takes a to the identity permutation ε). As usual, this means $a = \tau_k \ldots \tau_2 \tau_1$. Since the empty space moves around the puzzle and then eventually returns home, the number of moves must be even. To see why this is true, in Figure 32, the empty space must start in shaded box 16, and after each move it alternates the colour of the box it is in, and so if it returns to a shaded box it must have moved an even number of times. This means k is even, and so a is expressible as a product of an even number of transpositions. Therefore a is even.

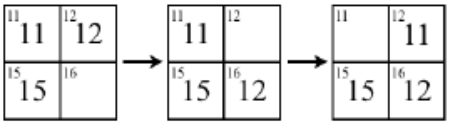
(ii) We wish to show every even permutation of the 15 tiles is obtainable through puzzle moves, starting from the solved-state. We will do this by showing we can obtain any 3-cycle of the tiles. This is enough to prove the theorem since any even permutation is expressible as a product of 3-cycles and if we can produce any 3-cycle then we can produce any product of them, through sequential moves, and therefore we can produce any even permutation.

We begin by observing we can produce the 3-cycle $\sigma=(11,12,15)$, by focussing on the bottom right corner of the puzzle:



The sequence of moves is: $(12,16)(11,12)(11,15)(15,16)$

Now that we have one 3-cycle σ , we will show that we can use σ to construct any other 3-cycle we want. From a solved puzzle, we pick any tile, for example $i\in\mathbb{Z}_{15}$. Then we move tiles 12 and 11 to boxes 16 and 12, respectively, by the move sequence $a=(12,16)(11,12)$.



Then using one of the two tours in Figure 34 we can move tile i to box 15, without disturbing the contents of boxes 12 and 16. Call this move sequence β.

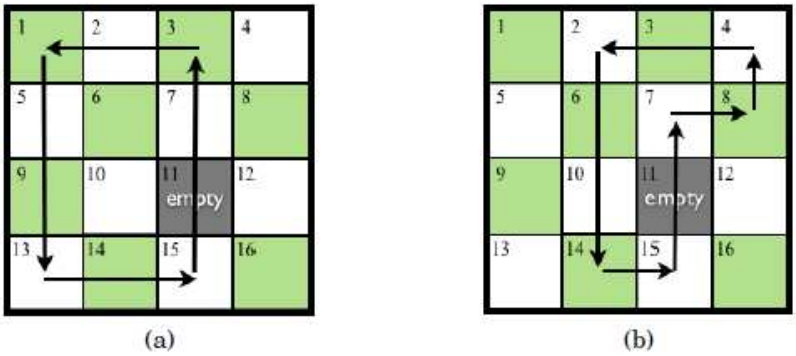
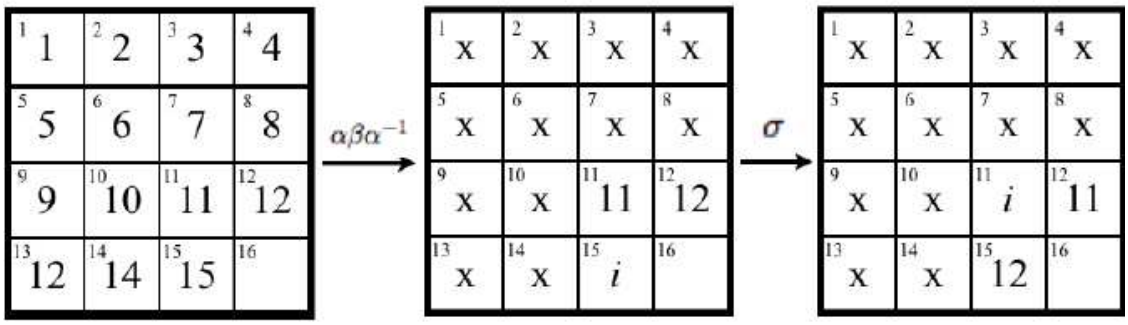


Figure 34. Tours for producing 3-cycles.

Applying a^{-1} then moves 11 and 12 back into their home positions. This puts the puzzle in the middle position in the following diagram, where the x's indicate these numbers may have moved around.

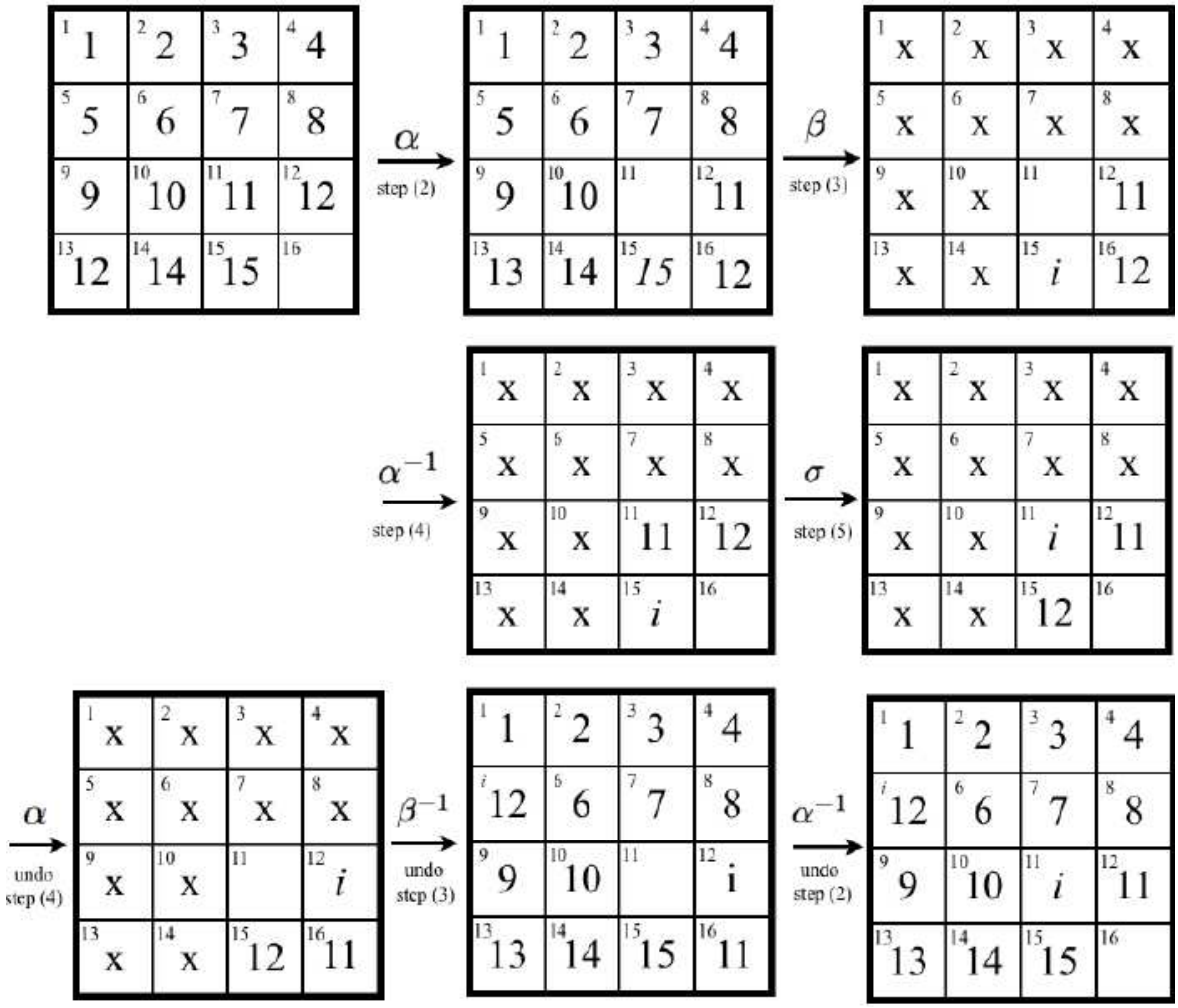


Applying the 3-cycle σ then moves tiles i, 11, and 12 around as indicated in the diagram. Now, we apply the inverse move sequence $(a\beta a^{-1})^{-1} = a\beta^{-1}a^{-1}$ and this takes everything back to where it was, except i stays in box 11, 11 stays in box 12, and 12 goes to box i. Therefore, we have created the 3-cycle $(11, 12, i)$, where i is any other tile we wish.

Summarizing:

1. we chose some tile i,
2. temporarily we hide tiles 11 and 12 in boxes 12 and 16,
3. we used one of the tours in Figure 34 to bring tile i to box 15, and this didn't disturb tiles 11 and 12 hidden in boxes 12 and 16
4. we moved 11 and 12 back out to their original positions
5. we applied the 3-cycle $\sigma = (11, 12, 15)$
6. then we reversed all the steps (4) to (2), thus taking everything home except 11,12, and i have been cycled

Here is an example (near the end we think of i as being 5 for concreteness).



So we now can construct any 3-cycle of the form $(11, 12, i)$, for any $i \neq 11, 12$.

Since $(11, 12, k)(11, 12, j) = (11, j)(12, k)$ we are able to put any tiles (j and k) in boxes 11 and 12, while leaving everything else in place. Moreover,

$$(11, j)(12, k)(11, 12, i)(11, j)(12, k) = (i, j, k),$$

where $i \neq j \neq k$ and $i, j, k \notin \{11, 12\}$. Therefore, we can produce any possible 3-cycle.

This completes the proof. □

The proof of the general solvability condition is a simple consequence of this specific case.

Proof of Theorem 7.1 - Part 2: Let a be the current permutation of the 15-puzzle. Move the empty space to box 16, then the new arrangement corresponds to the permutation

$$a^* = a\tau_1\tau_2...\tau_k$$

where $\tau_1\tau_2...\tau_k$ were the transpositions used to move the empty space to box 16. Since the empty space is now in box 16 then, by Theorem 7.1(Part 1), a^* is solvable if and only if it is an even permutation.

Let's think about when a^* is even. This follows from the way we defined the parities of the boxes. If the empty space was in an odd box, then it would have taken an odd number of transpositions to move it to box 16. That is, k would be odd. On the other hand, if the empty space was in an even box then k would be even, since it would have taken an even number of transpositions to move it to box 16. In either case, k is equal to the parity of the box the empty space was in. This is precisely the reason we defined the parity of a box as we did.

Now, putting it all together, a is solvable if and only if $a^* = a\tau_1\tau_2...\tau_k$ is even, which is equivalent to a and k having the same parity, which is equivalent to a and the location of the empty space having the same parity. This completes the proof. □

7.3 Strategy for solution of the 15-puzzle

We have essentially presented a strategy for solution in proving Theorem 7.1 . First we should move the empty space into box 16, then the resulting permutation is even, so we may express it as a product of 3-cycles. In practice, our typical method for doing this is first to express it as a product of transpositions, then group pairs of transpositions and express them as 3-cycle or pairs of 3-cycles. We can now use the technique outlined in Section 7.2 to produce each 3-cycle, one-by-one, by moving the desired tiles into the 11,12,15 boxes, performing the 3-cycle (11,12,15), then moving everything back again.

Though theoretically possible, and a perfectly sound way to prove the theorem, this makes for a completely inelegant way to solve the puzzle. Not to mention we should need to remember, or write down, the move sequence $a\beta a^{-1}$ since we should need to apply the inverse. This move sequence could be very long. Instead we will look for a more efficient solution, and one that doesn't require remembering any previously made moves.

Firstly, we should solve the puzzle by setting the tiles in their proper places, one-by-one, in numerical order. At some stages, it may be necessary to temporarily disturb placed pieces, but they shouldn't have to move too far out of place.

However, there are some tricky parts. For instance if 1,2,3 are all in place, but 4 is not, it will be necessary to disturb the previously placed pieces in order to get 4 in its proper place. Instead, before placing 3, join it with 4 to form a chain and bring the two of them into place together. Forming chains of tiles is a useful strategy.

Then getting the final few pieces in the proper places is of course tricky. But at this stage, making use of 3-cycles, as in the proof above, may be useful.

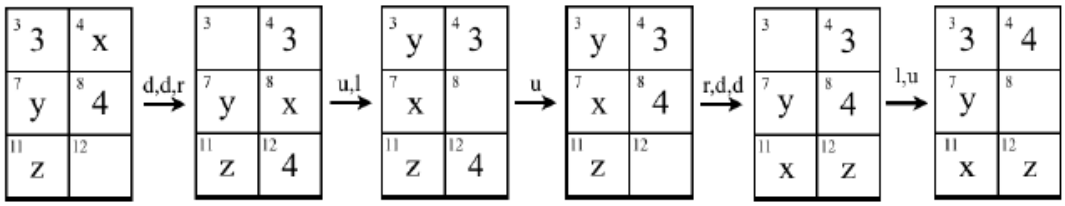
7.4 Solution of 15-puzzle

The following method depends on the virtual puzzle of Jaap Scherphuis [13]. It is a general method that works on any solvable configuration, and it could be extended to puzzles of sizes other than 4-by-4.

Phase 1: We check the solvability of the puzzle if it is not solvable we swap the last two tiles in order to get a solvable puzzle.

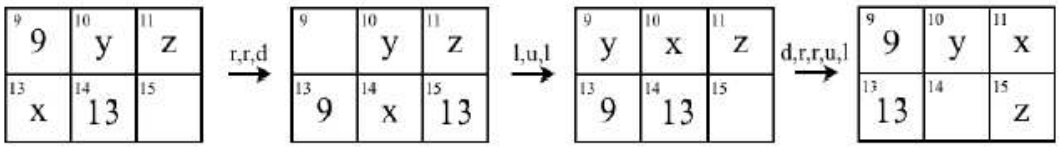
Phase 2: We solve the top row from left to right:

1. We find the next tile we want to place in position in the top row.
2. If it is not the last tile of the row, it is fairly easy to place correctly, by simply moving the tile in a certain direction and moving the other tiles around until the space is next to our tile on the side we want to move it to. Then we can move the tile.
3. If the last tile is not already in position, we bring it to the position directly below its correct spot, with the space directly below that. Then we move tiles in the following directions: down, down, right, up, left, up, right, down, down, left, up, as shown in the figure below. This should place the piece in position, however it does temporarily disturb the previously placed tile.



Phase 3: We solve the rest of the puzzle:

1. We use the technique in phase 2 to solve each row in turn, until there are only two rows left.
2. We rotate the puzzle a quarter turn to the right. The left column of the two rows becomes the top row now.
3. We use the technique in phase 2 to solve each row in turn, until there are only two rows left. This means there is only a 2x2 square left to solve. For example, the next figure shows how to get tile 12 in the correct place in the bottom left corner of the 4-by-4 version.



4. We move the pieces in the remaining 2x2 square around until one piece is positioned correctly, and the space is in the correct spot. The other two tiles should automatically be correctly positioned as well.

5. If there are two tiles that need to be swapped, then this cannot be done unless two other tiles are swapped as well. If there are two identical tiles somewhere in the puzzle, then we will have to swap them and solve the rest again.

7.5 Solution of 15-puzzle with GAP

In this section we present the implementation of the method for the solution of the 15-puzzle, described above, written in GAP programming language and tested with GAP version 4.7.8. If the reader of this thesis is interested for further practice and experimentation with this solution, he could copy and save the code that is presented below, in a wordpad and then after installation of GAP he could execute and test the solution. In section 1 we have already presented instructions for installation of GAP.

```
blankx:=0;;  #global variables
blanky:=0;;
hgh:=4;;
wid:=4;;
size:=hgh*wid;;

displaymatrix:=function(list)
    #display the puzzle
    local puzzle;
    puzzle:=[list{[1..wid]},list{[wid+1..2*wid]},list{[2*wid+1..3*wid]},list{[3*wid+1..4*wid]}];
    PrintArray(puzzle);
    Print("\n");
    return (puzzle);
end;



blank_piece:=function(puzzle)
    #find the blank piece
```

```
        local i,j;
        for i in [1..hgh] do
                for j in [1..wid] do
                        if puzzle[i][j]=size then blankx:=i;blanky:=j; fi;
                od;
        od;
        return();
end;




perm_parity:=function(list)
        local c,c1,i,j;

        #check permutation parity of puzzle
        c:=0;
        for i in [2..size] do
                for j in [1..i-1] do
                        if list[j]>list[i] then c:=c+1; fi;
                od;
        od;

        #check position of blank space, move to bottom right
        c1:=c + (wid)- blanky +(hgh)-blankx;

        return(c1);
end;




check_solution:=function(puzzle)
        #check if puzzle is solved
        local c,i,j;
        c:=0;
```

```
        for i in [1..hgh] do
                for j in [1..wid] do
                        c:=c+1;
                        if puzzle[i][j]<>c then return(false); fi;
                od;
        od;
        Print("15puzzle is solved","\n");
        return();
end;



domove:=function(puzzle,m)   # 0=right, 1=down, 2=up, 3=left
        local tmp;

        if m=0 then
                tmp:=puzzle[blankx][blanky-1];
                puzzle[blankx][blanky-1]:=size;
                puzzle[blankx][blanky]:=tmp;
                blanky:=blanky-1;
        elif m=1 then
                tmp:=puzzle[blankx-1][blanky];
                puzzle[blankx-1][blanky]:=size;
                puzzle[blankx][blanky]:=tmp;
                blankx:=blankx-1;
        elif m=2 then
                tmp:=puzzle[blankx+1][blanky];
                puzzle[blankx+1][blanky]:=size;
                puzzle[blankx][blanky]:=tmp;
                blankx:=blankx+1;
        elif m=3 then
                tmp:=puzzle[blankx][blanky+1];
                puzzle[blankx][blanky+1]:=size;
                puzzle[blankx][blanky]:=tmp;
                blanky:=blanky+1;
```

```
        fi;
        return(puzzle);
end;




movepiece:=function(puzzle,p,x,y)
        #moves piece p to position x,y without disturbing previously placed pieces

        #find position of piece p
        local yp,xp,i,j;
        for i in [1..hgh] do
                for j in [1..wid] do
                        if puzzle[i][j]=p then xp:=i;yp:=j; fi;
                od;
        od;
        Print("restore piece:",p,"\n");

        #move piece to same column  0=right, 1=down, 2=up, 3=left
        if yp<y and blankx=x then
                #move blank down if it might disturb solved pieces.
                puzzle:=domove(puzzle,2);
                PrintArray(puzzle); Print("\n"); fi;

        while yp>y do
                #move piece to left
                #First move blank to left hand side of it
                if blankx=xp and blanky>yp then
                        #if blank on wrong side of piece
                        if xp=hgh then puzzle:=domove(puzzle,1);
                        else puzzle:=domove(puzzle,2);
                        fi;  #then move it to other row
                PrintArray(puzzle); Print("\n");
                fi;
```

```
        while blanky>=yp do
                puzzle:=domove(puzzle,0); PrintArray(puzzle); Print("\n");
        od;  #move blank to column left of piece
        while blanky<yp-1 do
                puzzle:=domove(puzzle,3); PrintArray(puzzle); Print("\n");
        od;
        while blankx<xp do
                puzzle:=domove(puzzle,2); PrintArray(puzzle); Print("\n");
        od;  #move blank to same row as piece
        while blankx>xp do
                puzzle:=domove(puzzle,1); PrintArray(puzzle); Print("\n");
        od;
        puzzle:=domove(puzzle,3);      #move piece to left
        PrintArray(puzzle); Print("\n");
        yp:=yp-1;
od;

while yp<y do
        #move piece to right
        #First move blank to right hand side of it
        if blankx=xp and blanky<yp then
                if xp=hgh then  puzzle:=domove(puzzle,1);
                else puzzle:=domove(puzzle,2); fi;
        PrintArray(puzzle); Print("\n");
        fi;
        while blanky<=yp do
                puzzle:=domove(puzzle,3); PrintArray(puzzle); Print("\n");
        od;
        while blanky>yp+1 do
                puzzle:=domove(puzzle,0); PrintArray(puzzle); Print("\n");
        od;
        while blankx<xp do
                puzzle:=domove(puzzle,2); PrintArray(puzzle); Print("\n");
        od;
        while blankx>xp do
                puzzle:=domove(puzzle,1); PrintArray(puzzle); Print("\n");
```

```
        od;
        puzzle:=domove(puzzle,0);
        PrintArray(puzzle); Print("\n");
        yp:=yp+1;
od;

#Move piece up to same row     # 0=right, 1=down, 2=up, 3=left
while xp>x do
        if x<xp-1 then
                while blankx<xp-1 do
                        puzzle:=domove(puzzle,2); PrintArray(puzzle); Print("\n");
                od;
                if blanky=yp then
                        if yp=wid then puzzle:=domove(puzzle,0);
                        else puzzle:=domove(puzzle,3); fi;
                PrintArray(puzzle); Print("\n");
                fi;
                while blankx>xp-1 do
                        puzzle:=domove(puzzle,1); PrintArray(puzzle); Print("\n");
                od;
                while blanky<yp do
                        puzzle:=domove(puzzle,3); PrintArray(puzzle); Print("\n");
                od;
                while blanky>yp do
                        puzzle:=domove(puzzle,0); PrintArray(puzzle); Print("\n");
                od;
                puzzle:=domove(puzzle,2);
                PrintArray(puzzle); Print("\n");
        else
                if yp<>wid then
                        if blankx=xp and blanky=yp+1 then
                                puzzle:=domove(puzzle,1); PrintArray(puzzle);Print("\n");
                        fi;
                        if blankx=xp then
                                puzzle:=domove(puzzle,2); PrintArray(puzzle);Print("\n");
                        fi;
```

```
        while blanky<yp+1 do
            puzzle:=domove(puzzle,3); PrintArray(puzzle);Print("\n");
        od;
        while blanky>yp+1 do
            puzzle:=domove(puzzle,0); PrintArray(puzzle);Print("\n");

        od;
        while blankx>xp-1 do
            puzzle:=domove(puzzle,1); PrintArray(puzzle);Print("\n");
        od;
        while blankx<xp-1 do
            puzzle:=domove(puzzle,2); PrintArray(puzzle);Print("\n");
        od;
        puzzle:=domove(puzzle,0);
        puzzle:=domove(puzzle,2);
        PrintArray(puzzle); Print("\n");
    elif blankx<xp and blanky=yp then
        while blankx<xp do
            puzzle:=domove(puzzle,2); PrintArray(puzzle);Print("\n");
        od;
    else
        while blankx>xp+1 do
            puzzle:=domove(puzzle,1); PrintArray(puzzle);Print("\n");
        od;
        while blankx<xp+1 do
            puzzle:=domove(puzzle,2); PrintArray(puzzle);Print("\n");
        od;
        while blanky<yp do
            puzzle:=domove(puzzle,3); PrintArray(puzzle);Print("\n");
        od;
        while blanky>yp do
            puzzle:=domove(puzzle,0); PrintArray(puzzle);Print("\n");
        od;
        puzzle:=domove(puzzle,1);
        puzzle:=domove(puzzle,1);
        puzzle:=domove(puzzle,0);
```

```
                puzzle:=domove(puzzle,2);
                puzzle:=domove(puzzle,3);
                puzzle:=domove(puzzle,2);
                puzzle:=domove(puzzle,0);
                puzzle:=domove(puzzle,1);
                puzzle:=domove(puzzle,1);
                puzzle:=domove(puzzle,3);
                puzzle:=domove(puzzle,2);
                PrintArray(puzzle); Print("\n");
            fi;
        fi;
        xp:=xp-1;
od;

while xp<x do
        #move piece downwards
        #First move blank below tile
        if blanky=yp and blankx<xp then
            if yp=wid then puzzle:=domove(puzzle,0);
            else puzzle:=domove(puzzle,3); fi;
        PrintArray(puzzle); Print("\n");
        fi;
        while blankx>xp+1 do
            puzzle:=domove(puzzle,1); PrintArray(puzzle);Print("\n");
        od;
        while blankx<xp+1 do
            puzzle:=domove(puzzle,2); PrintArray(puzzle);Print("\n");
        od;
        while blanky<yp do
            puzzle:=domove(puzzle,3); PrintArray(puzzle);Print("\n");
        od;
        while blanky>yp do
            puzzle:=domove(puzzle,0); PrintArray(puzzle);Print("\n");
        od;
        puzzle:=domove(puzzle,1);
        PrintArray(puzzle); Print("\n");
```

```
            xp:=xp+1;
        od;

        return(puzzle);
end;



solve:=function()

        local list,puzzle,listblank,i,j,tmpi,tmpi2,tmpj,tmpj2,c1,r,rr,c,solution;

        # create a random list for the puzzle
        list:=Shuffle([1..size]);

        Print("Random 15puzzle for solution","\n");
        puzzle:=displaymatrix(list);

        listblank:=blank_piece(puzzle);

        c1:=perm_parity(list);

        if c1 mod 2 <> 0 then
            #swap the last two pieces 14,15
            Print("swap last two pieces 14,15 in order to be solvable","\n");
            tmpi:=0;tmpi2:=0;tmpj:=0;tmpj2:=0;

            for i in [1..hgh] do
                for j in [1..wid] do
                    if puzzle[i][j]=size-2 then tmpi:=i;tmpj:=j;fi;
                    if puzzle[i][j]=size-1 then tmpi2:=i;tmpj2:=j;fi;
                od;
            od;
            puzzle[tmpi][tmpj]:=size-1;
            puzzle[tmpi2][tmpj2]:=size-2;
            PrintArray(puzzle);
```

```
        Print("\n");
fi;



#restore top rows
rr:=0;
for r in [1..hgh-2] do
        for c in [1..wid] do
                puzzle:=movepiece(puzzle,rr+c,r,c);
        od;
        rr:=rr+wid;
od;

Print("restore left columns","\n");

#restore left columns
for c in [1..wid-2] do
        #restore top tile of column.
        puzzle:=movepiece(puzzle,rr+c,hgh-1,c);

        #restore bottom tile of column
        if blanky=c then puzzle:=domove(puzzle,3); PrintArray(puzzle); Print("\n"); fi;
        if puzzle[hgh][c]<>rr+c+hgh then
                puzzle:=movepiece(puzzle,rr+c+hgh,hgh,c+1);
                if blankx<>hgh then     #0=right, 1=down, 2=up, 3=left
                        #A_X or AX_
                        #XBX    XBX
                        if blanky=c+1 then
                                puzzle:=domove(puzzle,3);
                                PrintArray(puzzle); Print("\n");
                        fi;
                        puzzle:=domove(puzzle,2);
                        PrintArray(puzzle); Print("\n");
                fi;
                #AXX
                #XB_
                while blanky>c+2 do
```

```
                puzzle:=domove(puzzle,0);
                PrintArray(puzzle); Print("\n");
            od;
            puzzle:=domove(puzzle,0);
            puzzle:=domove(puzzle,0);
            puzzle:=domove(puzzle,1);
            puzzle:=domove(puzzle,3);
            puzzle:=domove(puzzle,2);
            puzzle:=domove(puzzle,3);
            puzzle:=domove(puzzle,1);
            puzzle:=domove(puzzle,0);
            puzzle:=domove(puzzle,0);
            puzzle:=domove(puzzle,2);
            puzzle:=domove(puzzle,3);
            PrintArray(puzzle); Print("\n");
        fi;
od;

Print("restore last 2x2 square","\n");

#last 2x2 square
if blanky<wid then puzzle:=domove(puzzle,3); PrintArray(puzzle); Print("\n"); fi;
if blankx<hgh then puzzle:=domove(puzzle,2); PrintArray(puzzle); Print("\n"); fi;
rr:=size-wid-1;
if puzzle[hgh-1][wid-1]=rr then
        if puzzle[hgh-1][wid]=rr+1 then
            if blanky=wid-1 then
                puzzle:=domove(puzzle,3); PrintArray(puzzle); Print("\n");
            fi;
        else
            puzzle:=domove(puzzle,1); PrintArray(puzzle); Print("\n");
            puzzle:=domove(puzzle,0); PrintArray(puzzle); Print("\n");
            puzzle:=domove(puzzle,2); PrintArray(puzzle); Print("\n");
            puzzle:=domove(puzzle,3); PrintArray(puzzle); Print("\n");
        fi;
fi;
if puzzle[hgh-1][wid-1]=rr+1 then
```

```
            puzzle:=domove(puzzle,1); PrintArray(puzzle); Print("\n");
            puzzle:=domove(puzzle,0); PrintArray(puzzle); Print("\n");
            puzzle:=domove(puzzle,2); PrintArray(puzzle); Print("\n");
            puzzle:=domove(puzzle,3); PrintArray(puzzle); Print("\n");
        fi;
        if puzzle[hgh-1][wid-1]=rr+wid then
            puzzle:=domove(puzzle,0); PrintArray(puzzle); Print("\n");
            puzzle:=domove(puzzle,1); PrintArray(puzzle); Print("\n");
            puzzle:=domove(puzzle,3); PrintArray(puzzle); Print("\n");
            puzzle:=domove(puzzle,2); PrintArray(puzzle); Print("\n");
        fi;

        solution:=check_solution(puzzle);

        return();
end;

15puzzle:=solve();
```

8 Conclusion

In this thesis we used a collection of puzzles which had a common theme: the pieces of the puzzle could be rearranged or permuted, and the goal was to restore the pieces back to their original positions. We have shown that all of these various puzzles known as permutation puzzles are so mathematically rich and they all could be modeled and represented using permutations. As we have seen permutations can describe the puzzles current position and also the move sequence applied to the puzzle. Also we have shown that every permutation can be decomposed as a product of 2-cycles or 3-cycles and this decomposition can be connected to the solvability of puzzles, as we have seen with Swap puzzle and 15-puzzle. For this investigation on permutation puzzles and also for the solution of the 15-puzzle which we presented, we used GAP(version 4.7.8).

9 References

1. The GAP Group, GAP – Groups, Algorithms, and Programming, Version 4.7.8; 2015. (http://www.gap-system.org)

2. https://en.wikipedia.org/wiki/GAP_(computer_algebra_system)

3. David Joyner. "Adventures in Group Theory: Rubik's Cube, Merlin's Machine, and Other Mathematical Toys", second edition, 2008.

4. Jamie Mullholand. Courses website for Permutation Puzzles, Simon Fraser University http://www.sfu.ca/~jtmulhol/math302/lectures.html

5. J. Scherphuis. Jaap's puzzle page. http://www.jaapsch.net/puzzles/topspin.htm, 2016

6. Patent US 4,871,173 by Ferdinand Lammertink for Oval Track puzzle http://www.jaapsch.net/puzzles/patents/us4871173.pdf

7. Patent US 507,215 by William Churchill for Hungarian Rings puzzle http://www.jaapsch.net/puzzles/patents/us507215.pdf

8. J. Scherphuis. Jaap's puzzle page. http://www.jaapsch.net/puzzles/rings.htm, 2016

9. Patent HU 170,062 by Ernö Rubik for Rubik's Cube puzzle http://www.jaapsch.net/puzzles/patents/hu170062.pdf

10. J. Scherphuis. Jaap's puzzle page. http://www.jaapsch.net/puzzles/cube3.htm, 2016

11. V-Cubes worldwide patent by Panagiotis Verdes http://www.jaapsch.net/puzzles/patents/wo2004103497.pdf

12. David Singmaster. "Notes on Rubik's Magic Cube", April 1981

13. J. Scherphuis. Jaap's puzzle page. http://www.jaapsch.net/puzzles/fifteen.htm, 2016

14. K. Conrad. The 15-puzzle (and rubik's cube). notes, 2008.

15. A. F. Archer. A modern treatment of the 15-puzzle. American Mathematical Monthly, 106(9):793-799, 1999.

16. J. Slocum and D Sonneveld. The 15-Puzzle: How it Drove the World Crazy. The Slocum Puzzle Foundation, Beverly Hills, CA, 2006.

17. J.O. Kiltinen. Oval Track and Other Permutation Puzzles: And Just Enough Group Theory to Solve Them. The Mathematical Association of America, New York, 2003.

18. J.O. Kiltinen. How few transpositions suffice? ... you already know! Mathematics Magazine, 67(1):45-47, 1994.

19. U. Dudley. Elementary Number Theory. Dover Publications, 2nd edition, 2008.

20. C. Bandelow. Inside Rubik's Cube and Beyond. Birkhauser, Boston, 1982.

21. A. H. Frey Jr. and D. Singmaster. Handbook of Cubik Math. Enslow Publishers, New Jersey, 1982.

22. J. A. Gallian. Contemporary Abstract Algebra. DC Heath and Company, Lexington, 1994.

23. J.G. Nourse. The Simple Solution to Rubik's Cube. Bantam Books, Toronto, 1981.

24. J. Slocum, D. Singmaster, D. Gebhardt, W. H. Huang, and G. Hellings. The Cube: The Ultimate Guide to the World's Best Selling Puzzle. Black Dog & Leventhal Publishers, Inc., New York, 2009.

25. N. Reilly. Applied algebraic systems. course notes, 1998.

26. GAP - Reference Manual, Release 4.8.2, 20-Feb-2016, http://www.gap-system.org/Manuals/doc/ref/chap0.html

27. Alexander Hulpke with contributions by Kenneth Monks and Ellen Ziliak, "Abstract Algebra in GAP", version January 2011

28. The history of the 15puzzle, http://www.hc11.demon.nl/15puzzle/15puzzen.htm

29. Reference sheet: "Computer Algebra Software: Mathematica, SymPy, GAP, Pari/GP", 2016, http://hyperpolyglot.org/computer-algebra

30. Online community, http://mathoverflow.net/questions